Universidad Carlos III de Madrid

TESIS DOCTORAL

Elementos Locales en Conjuntos de Clasificadores Diseñados por "Boosting"

Autor:

Efraín T. Mayhua López

Directores:

Dr. Aníbal R. Figueiras Vidal

Dra. Vanessa Gómez Verdejo

Dpto. de Teoría de la Señal y Comunicaciones

Leganés, Abril 2013

TESIS DOCTORAL

ELEMENTOS LOCALES EN CONJUNTOS DE CLASIFICADORES DISEÑADOS POR "BOOSTING"

Autor:
EFRAIN T. MAYHUA LOPEZ

Directores:
Dr. ANÍBAL R. FIGUEIRAS VIDAL
Dra. VANESSA GÓMEZ VERDEJO

Firma del Tribunal Calificador:

Firma

Presidente:

Vocal:

Secretario:

Calificación:

Leganés, de de

RESUMEN

Aunque los diseños bloque de Máquinas de Aprendizaje son en general una herramienta poderosa para resolver problemas de inferencia, en situaciones complicadas pueden resultar insuficientes. Para solventar esta limitación cabe considerar la combinación de diferentes máquinas (simples) de modo que se forme un conjunto capaz de resolver mejor el problema en cuestión, proporcionando, además, un diseño más sencillo y más fácilmente comprensible. Entre los conjuntos destacan, por sus sencillos principios conceptuales y sus contrastadas buenas prestaciones, los métodos de Boosting, y, especialmente, el algoritmo "Real AdaBoost" (RAB).

A pesar de que las características mencionadas han llevado a que el empleo de los métodos de Boosting sea cada vez más frecuente, su diseño está limitado, por un lado, porque los aprendices que componen el conjunto se combinan linealmente, y, por otro lado, al uso de clasificadores base de tipo global o, de ser locales, con limitada capacidad expresiva. Pues bien: el objetivo de esta Tesis Doctoral es mejorar las prestaciones de los métodos de Boosting introduciendo elementos locales con estructuras de núcleos en los aprendices o en su fusión, con el objeto de paliar las citadas limitaciones.

Así, en primer lugar, se presenta y aplica un esquema de fusión controlado por puertas similares a las empleadas en las Mezclas de Expertos. Este nuevo modelo de fusión utiliza una puerta entrenable compuesta por un cuerpo con estructuras de núcleos sencillos y un vector de pesos de salida que se ajusta paso a paso minimizando la función de coste del RAB, sin necesidad de realizar modificaciones en el modo de entrenar los aprendices. Los resultados obtenidos indican que estos esquemas permiten frecuentemente alcanzar mejores prestaciones con baja carga computacional en operación, sí bien a cambio de un sensible incremento de carga computacional en diseño debido a la necesidad de determinar mediante validación cruzada los valores de los parámetros propios de la puerta.

En segundo lugar, se introduce un nuevo procedimiento que permite el uso de Máquinas de Vectores Soporte (SVMs) como aprendices de conjuntos construidos

mediante Boosting. Se diseñan estos aprendices aplicando un procedimiento de submuestreo en una SVM con Programación Lineal, lo que posibilita el uso de una matriz núcleo submuestreada por filas, con la consiguiente flexibilidad a la hora de controlar la diversidad y complejidad de este tipo de aprendices. Esta combinación de modificaciones de diseño ha dado como resultado conjuntos con estructuras de núcleos reutilizables por los diferentes aprendices, característica que ha permitido conseguir máquinas compactas de muy buenas prestaciones y con una carga computacional de operación similar a la de una SVM individual.

El trabajo se cierra con una revisión de las aportaciones y sugerencias de líneas de investigación abiertas.

ABSTRACT

Although the block design of learning machines is, in general, a powerful tool to solve inference problems, it can be insufficient in complex scenarios. In order to solve this limitation, it is necessary to consider combinations of different (simple) machines, that form an ensemble which is not only able to solve the problem at hand but also to provide a simpler and somewhat more understandable design. Among different ensemble building techniques, Boosting methods and, in particular, the "Real AdaBoost" (RAB) algorithm prevail due to their simple conceptual principles and their well-known good performance.

Despite the above-mentioned characteristics have caused an increasing use of Boosting methods, their designs are limited. A first limitations comes from the linear combination of the ensemble learners, and, on the other hand, by the use of global base classifiers or, when including traditional local learners, by their limited expressive power. The objective of this Ph.D. Thesis is just to improve the performance of Boosting methods by introducing local elements with a kernel-based structure either in the learners or in their fusion, with the purpose of reducing the effect of the mentioned limitations.

In this way, firstly, a method based on including fusion gates similar to those used in Mixtures of Experts, is presented and applied. This new fusion model employs a trainable gate consisting of a set of simple kernel structures and an output weight vector which is step-by-step adjusted so that the RAB cost function is minimized, without the need of modifying the learner training. Obtained results evidence that these schemes frequently achieve performance improvements with a low operation computational load, but with a significant computational burden increment in the design process due to the need of determining the gate parameters by a cross validation procedure.

Secondly, a new procedure to allow the use of Support Vector Machines (SVM) as base learners of Boosting ensembles is presented. These learners are designed by applying a sub-sampling procedure in a Linear Programming SVM, making possible

the use of a kernel matrix subsampled by rows and, therefore, providing flexibility to control the diversity and complexity of this kind of learners. These design modifications have provided ensembles with kernel structures which are reusable by the different learners. This relevant characteristic allows to achieve compact machines with accurate performances and a computational load similar to that of a single SVM.

This document concludes with a review of the main contributions and some suggestions about new research lines arising from this work.

AGRADECIMIENTOS

Esta Tesis Doctoral es el resultado de cinco años de trabajo y esfuerzo que han supuesto todo un reto personal. Entre los momentos de soledad, desesperación y euforia he encontrado instantes de calma que me han permitido disfrutar del mundo de la investigación. Durante todo este tiempo, han sido muchas las personas que me han acompañado, han sabido compartir conmigo mis éxitos ayudándome a afrontar mis fracasos. Por lo tanto, son muchas las gracias que debo dar y desde aquí, quiero transmitir mi agradecimiento a todos ellos:

A mis Directores de Tesis: el Dr. Aníbal R. Figueiras Vidal y la Dra. Vanessa Gómez Verdejo. Gracias Aníbal por darme la oportunidad de pertenecer a tu grupo de investigación, hacer un espacio en tu apretada agenda y compartir conmigo parte de tu tiempo, introducirme en un mundo desconocido para mí, dándome así, la oportunidad de aprender mucho más de lo que esperaba. Gracias Vanessa por tener la paciencia suficiente de aguantar mis divagaciones, estar dispuesta siempre a ayudarme con tus sugerencias y, sobre todo, alentarme a seguir adelante con tu apoyo constante.

A mis compañeros del laboratorio 42C03, quienes se han encargado de hacerme sentir como en casa. Los inolvidables compañeros de INDRA Guillermo, Ivan y Javi, ha sido un placer compartir espacio de trabajo con ustedes. Mis grandes amigos, los más antiguos: Luis, Soufiane, Roberto, Sergio y Adil, gracias por todo, tenemos el sello de habernos subido en una tabla en el mar del África gracias a la pericia de Sergio. Los más nuevos: Mariléa, Juan José, Fernando, Rafa, Ricardo y Anas. Gracias a todos por los cafés, los mixtos con huevo y las barritas con tomate. Dentro de este grupo, como olvidarse de Martha, siempre con los comentarios acertados dando vida al grupo.

A los Doctores: Miguel, siempre con cosas nuevas acorde a su brillantes, y Emilio Gedeón, un gran tipo que inspira mucha confianza.

A los profesores del Dpto. de Teoría de la Señal y Comunicaciones, gracias por su apoyo y su disposición a resolver las dudas que me envergaban y no me dejaban

dormir en los primeros años del Máster.

A las señoras de la limpieza, Encarnación y Fidela. Han sido mi reloj de arena, cada vez que venían a fregar el piso del laboratorio significaba que ha pasado una semana más y una semana menos.

A mis compañeros de la liga 300!!. El gran Giorgio, que a estas alturas debe ser algo así como un Rey en Patras. Luca, siempre lleno de excentricidades. Los colombianos, Carlos, Diego, Julio, Wilton, Jimmy, Camilo, Jair, Guillermo,..., y como olvidarse de los Electrónicos, Dani, los David, Pedrito,..., gracias por enseñarme a jugar al fútbol, fue el complemento perfecto.

A mis alumnos y compañeros de trabajo de la Universidad Católica San Pablo (UCSP) en Arequipa, que no los nombraré porque no podría perdonarme el olvido de alguien, gracias a todos ustedes por siempre alentarme a seguir adelante.

A German Chávez y Alonso Quintanilla, Rector y ProRector de la UCSP. Gracias por confiar en mí, darme la oportunidad de ser parte del proyecto UCSP, entregarme las facilidades para sentirme a gusto en la universidad y sobre todo, por su amistad. También, debo mucho agradecimiento a Jafeth, Gonzalo, Erika, Jimmy, Samuel, en sus nombres, gracias a todos los profesores del Programa Profesional de Ingeniería de Telecomunicaciones de la UCSP, su amistad es algo que siempre voy a valorar.

A mis amigos de siempre, José Maldonado y Ángel Escalante, gracias por los grandes momentos que hemos compartido y seguro seguiremos compartiendo.

A mis amigos de la UNSA, Carlos, Rolo, Manuel, Julio, Edwin, Gino, Juan, Carlos Mayorga,..., simplemente gracias por su amistad.

A mi gran amigo Jorge Tamayo, como olvidarse de la gran aventura de easynet, la fiesta a lo bikingo en Juliaca y tantas otras cosas. A Leo por ser parte de este equipo lleno de sueños.

A los integrantes de la familia Argüelles Bendezú (Manuel, Rosa, Patricia, Juan Manuel, Irene) muchas gracias por todo el cariño que siempre han volcado hacia mi persona y haberse encargado de mis asuntos en varias ocasiones. Gracias por todo hermanita Irene y en especial a Patricia, mi gran amiga y compañera. Gracias Paty

por todos tus consejos, siempre han sido acertados, gracias por visitarme a Madrid y Barcelona, siempre te has hecho presente en el momento adecuado. Sobre todo, gracias por los maravillosos días entre la navidad del 2012 e inicio del año 2013, me he sentido muy a gusto. Necesitaba de las buenas vibras de una persona como tú para recuperar la confianza en mí mismo, tomar fuerzas y emprender el viaje a Madrid y dar por concluido esta Tesis.

A dos personas que significan mucho para mí: Paola (una mujer admirable) y Adriana (una hija ejemplar), por las que siento mucho amor las quiero y estimo mucho. Gracias Paolita por entregarme tanto amor y haberme acompañado desde la distancia durante todos estos años que me ha demorado terminar esta Tesis Doctoral. Sin tu compañía seguramente todo habría sido más complicado. Siento mucho que al final no podamos disfrutar juntos de este resultado. Siempre estas presente en mí y espero que algún día consigas entender y creer en este aprendiz de investigador.

Adrinita, ahora sí, ya puedo responder a tu pregunta: ¿Cuándo vas a venir? Y será mucho antes de tu cumpleaños. Siempre serás la hija que me gustaría tener, gracias por esas horas interminables de conversación, haberme confiado tus secretos y deseos (deseos que estoy seguro que los harás realidad). Gracias por todos tus escritos, las tarjetas y presentes (hechos por ti y muy originales por cierto) que me has entregado, los guardo con mucho cariño.

Siempre recordare con nostalgia todas las cosas que hacíamos los tres juntos, nuestro viaje a Chanchamayo (nuestras palabras Asháninca: pasonki y avíro), comprando chuches en Lima (todo para Paola y nada para nosotros), los juegos (que me ganaban asociándose y con trampa), aprendiendo costura (el disfraz de mariquita que solo lo vi por foto), cocinando (pollo al sillao), haciendo las tareas del colegio o ir por una "peluka loca" (que por cierto te debo una Adrianita), sobre todo, aquellos días en los que nuestra única preocupación era caminar sin pisar las líneas del suelo. Los echo de menos y echaré de menos mucho.

A mis nueve hermanos. César, que seguro desde donde estas iluminas mi camino, espero no haberte defraudado, gracias por enseñarme que después del colegio esta la

universidad, ello ha hecho que yo me empeñe en hacer la Tesis. Mi hermana Esther, gracias por todo el sacrifico que haces por nosotros. José, tu apoyo y entrega por la familia son invalorables. Ciro, siempre lleno de proyectos, sigue adelante y nunca te canses. Gracias Ciro por hacerme padrino de Irina. Hidel y Yinaldi, gracias por traer vida a la casa: Luciana y Fabián. Moisés, lleno de talento que aún no lo quieres explotar. Margot, entregas tú vida en servicio de los demás, cosa que para ti es solo un gesto, pero para nosotros simplemente no tendríamos la capacidad ni fuerzas para hacerlo. Juan Carlos, el menor de todos, sigue adelante hasta conseguir tus objetivos. Gracias a todos ustedes hermanos, por todos los momentos, las pequeñas y grandes cosas que hemos compartido y seguiremos compartiendo.

Por último, a mis padres. A mi padre Teófilo, que nos dejó mientras yo intentaba sacar adelante el Máster, nunca sentí tanto miedo de enfrentarme al dolor de tú partida, al final me deje ganar por ese miedo. Gracias padre por todos tus enseñanzas, sobre todo tu arte —la sastrería— que me ha permitido pagarme parte de la secundaria y la universidad.

A mi madre Antonia, en tus manos están grabados todo el esfuerzo que has entregado por nosotros, tus ganas de aprender a leer y escribir han hecho de mi lo que soy. Gracias por tu terquedad y sacarme de la sierra para enviarme a un colegio en la ciudad, sin ello, nada de esto habría sido posible.

Esta Tesis es el resultado de la formación que ustedes me han dado y como retribución a parte de ello, éste trabajo, lo dedico a ustedes.

Gracias a todos.

" *A un hombre creativo*
le motiva el deseo de lograr,
no el de vencer a otros"

Ayn Rand,
1905 -1982

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Este primer capítulo estable el contexto en que se desarrolla la Tesis, presentando algunos conceptos fundamentales del aprendizaje máquina, con especial atención a las técnicas que motivan las contribuciones aquí incluidas. Inicialmente, nos centramos en la teoría de la decisión (clasificación). Seguidamente, se revisan los principales algoritmos de clasificación máquina como preámbulo a los conjuntos de máquinas de aprendizaje. A continuación, se revisan los procedimientos para construir clasificadores conjuntos, destacando los métodos de Boosting.

Tras lo anterior, se expone la motivación y los objetivos concretos perseguidos en este trabajo, para finalizar el capítulo con la descripción del contenido de esta Tesis.

1.1. El problema de decisión o clasificación

El problema de decisión en el formato de test de hipótesis plantea cómo elegir entre un cierto número de hipótesis, $\Omega = \{\omega_1, \ldots, \omega_J\}$, a la vista de un dato, muestra o instancia, $\mathbf{x} = [x_1, \ldots, x_d]^\top$ (con d medidas de atributos) relacionado con ellas. Se soluciona estableciendo una función capaz de determinar (predecir) la etiqueta (hipótesis) de instancias no conocidas. Si las hipótesis son la pertenencia a clases, el problema se llama de clasificación.

Cuando se conoce la estructura estadística del problema, suele recurrirse a métodos analíticos [Van Trees, 2004, Barkat, 2005, Poor, 2010] para su resolución. Entre ellos se incluye la Teoría Bayesiana de la Decisión. En este caso, se considera λ_{ji} como el coste en que se incurre al asignar $\mathbf{x}$ a la j-ésima clase ω_j cuando en realidad es miembro de la i-ésima clase ω_i, y $P(\omega_i|\mathbf{x})$ es la probabilidad que la muestra $\mathbf{x}$ sea un miembro de la i-ésima clase. El *coste medio* asociado con aceptar ω_j es

$$R(\omega_j|\mathbf{x}) = \sum_{i=1}^{J} \lambda_{ji} P(\omega_i|\mathbf{x}) \tag{1.1}$$

y es también conocido como el *riesgo condicional* [Duda et al., 2001]. Cuando deseamos clasificar una observación en particular, $\mathbf{x}$, podemos proceder mediante la selección de la clase que lo minimiza (decisión de mínimo riesgo)

$$j^* = \arg \min_j R(\omega_j|\mathbf{x}). \tag{1.2}$$

Una función de coste de especial interés es el denominado coste *simétrico básico* o función de coste *cero-uno*,

$$\lambda_{ji} = \begin{cases} 0, & \text{si } \; j = i \\ 1, & \text{si } \; j \neq i \end{cases} \tag{1.3}$$

donde $j, i = 1, \ldots, J$. Esta función no penaliza las decisiones correctas y asigna un coste unitario a cualquier error. En este caso, el riesgo condicional puede expresarse como

$$R(\omega_j|\mathbf{x}) = \sum_{i=1}^{J} \lambda_{ji} P(\omega_i|\mathbf{x}) = \sum_{i \neq j} P(\omega_i|\mathbf{x}) = 1 - P(\omega_j|\mathbf{x}) \tag{1.4}$$

donde $P(\omega_j|\mathbf{x})$ es la probabilidad a posteriori de la clase j, y, por lo tanto, la ecuación (1.2) se puede reducir a

$$j^* = \arg \max_j P(\omega_j|\mathbf{x}) \tag{1.5}$$

lo que se conoce como criterio MAP (Máximo a Posteriori).

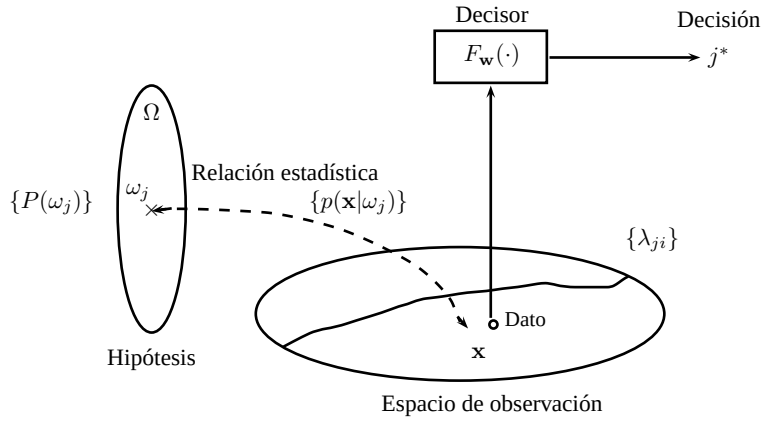


Figura 1.1: Visión analítica de la decisión

Para aplicar la reglas del decisor de mínimo riesgo (1.2) o el decisor MAP (1.5) es necesario conocer las verdaderas probabilidades a posteriori $\{P(\omega_j|\mathbf{x})\}_{j=1}^{J}$. En la práctica, no es habitual el acceso directo a esta información estadística y suele recurrirse a su cálculo a partir de la verosimilitud $p(\mathbf{x}|\omega_j)$ y la probabilidad a priori de cada hipótesis $P(\omega_j)$, haciendo uso del teorema de Bayes

$$P(\omega_j|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_j)P(\omega_j)}{\sum_{j=1}^{J} p(\mathbf{x}|\omega_j)P(\omega_j)} \tag{1.6}$$

En otras palabras, la regla de Bayes muestra cómo la observación de los valores de las medidas de $\mathbf{x}$ cambia la probabilidad a priori $P(\omega_j)$ por la probabilidad a posteriori $P(\omega_j|\mathbf{x})$. Por este motivo, a los decisores diseñados mediante (1.2) ó (1.5) se les llama decisores Bayesianos (véase la Figura 1.1).

El problema de clasificación también puede ser visto como la implementación de un conjunto de funciones discriminantes, $f_i(\mathbf{x}), \quad i = 1, \ldots, J$, de tal manera que el clasificador asigna la muestra $\mathbf{x}$ a la clase j si

$$f_j(\mathbf{x}) > f_i(\mathbf{x}) \quad \forall i \neq j \tag{1.7}$$

De este modo, el clasificador selecciona la clase correspondiente a la función discriminante con mayor valor.

La relación directa con la regla de decisión de mínimo riesgo se da cuando la función discriminante se corresponde con el riesgo condicional, es decir

$$f_j(\mathbf{x}) = R(\omega_j|\mathbf{x}) \tag{1.8}$$

Para el caso del decisor MAP, las cosas podrían simplificarse aún más mediante la adopción de $f_j(\mathbf{x}) = P(\omega_j|\mathbf{x})$, de manera que la función discriminante máxima es la que corresponde a la probabilidad a posteriori máxima. En todo caso, una decisión divide el espacio de características en J regiones de decisión, que se encuentran separadas por las denominadas fronteras de decisión.

Sin embargo, en muchas ocasiones no sólo no se conoce la probabilidad a posteriori, sino que tampoco se conoce la expresión analítica de la verosimilitud para poder aplicar el teorema de Bayes. En tales situaciones, se puede recurrir a métodos de estimación de funciones de densidad de probabilidad para estimar la verosimilitud a partir de muestras disponibles. Estos métodos se clasifican en tres tipos:

- Paramétricos: se asume un modelo analítico conocido para la función de densidad de probabilidad y se estiman sus parámetros a partir de las observaciones disponibles. Los enfoques más comunes para aprender los parámetros del modelo asumido son el de máxima verosimilitud (ML, "Maximum Likelihood") y los métodos basados en inferencia Bayesiana. Este tipo de métodos presenta inconvenientes cuando se asume un modelo incapaz de proporcionar una buena representación de la densidad real, dando lugar a clasificadores de bajas prestaciones.

- No paramétricos: en este caso no se asume un modelo analítico en particular, sino que se considera un modelo general capaz de aproximar cualquier tipo de función de densidad y se ajustan sus parámetros de acuerdo con las observaciones disponibles. El principal inconveniente de esta aproximación está en el número de parámetros del modelo, que crece con el número de las observaciones, implicando altas demandas de memoria y cómputo. En esta familia destacan

el método de los K vecinos más próximos (K-NN, "K-Nearest Neighbours") y los modelos basados en ventanas de Parzen.

- Semiparamétricos: estos métodos emplean modelos flexibles (típicamente combinaciones convexas) para aproximar las funciones de distribución. De este modo permiten conseguir buenas aproximaciones utilizando pocos parámetros y reducen el coste computacional. Entre este tipo de aproximaciones destacan los Modelos de Mezcla de Gaussianas (GMMs, "Gaussian Mixture Models"), cuyos parámetros se optimizan empleando el algoritmo de Esperanza-Maximización (EM, "Expectation-Maximization") [Dempster et al., 1977].

Respecto a las probabilidades a priori $P(\omega_j)$, o bien están disponibles de antemano a través de algún conocimiento de la naturaleza del problema estudiado, o se suelen calcular como las proporciones de cada clase en el conjunto de datos de entrenamiento (frecuencia relativa). Una discusión más amplia de todos estos métodos se encuentra en [Duda et al., 2001, MacKay, 2002, Bishop, 2006, Barber, 2012].

El clasificador que minimiza (1.1) no tiene por qué obtenerse únicamente a partir de la estimación de la función de densidad de probabilidad, sino que se puede derivar de un principio inductivo, dando lugar al diseño de clasificadores bajo el enfoque máquina, que parte de un conjunto de observaciones representativas del problema. En la siguiente sección se abordan los principales métodos máquina para clasificación.

1.2. Clasificación máquina

Como ya se ha indicado, la teoría de decisión (clasificación) estadística se sustenta en el supuesto de que se dispone de un modelo estadístico de la dependencia entre la pertenencia a clases y las observaciones. Dado que en la práctica este supuesto casi nunca se cumple, el diseño del clasificador debe abordarse a partir de una colección de datos representativos del problema. Es decir, se dispone de un conjunto de muestras etiquetadas, $\left\{\mathbf{x}^{(l)}, y^{(l)}\right\}_{l=1}^{L}$, que han sido generadas a partir de una distribución

desconocida y de manera independiente, con las cuales se pretende diseñar un clasificador que implemente una función $f(\mathbf{x})$ de mínima probabilidad de error, de tal forma que el mejor clasificador será aquel que minimice $R_{emp}(\mathbf{w})$,

$$R_{emp}(\mathbf{w}) = \frac{1}{L}\sum_{l=1}^{L} B(y^{(l)}, f(\mathbf{x}^{(l)}, \mathbf{w})) \tag{1.9}$$

donde $B(\cdot)$ es la función de coste y $\mathbf{w}$ el parámetro (vectorial) cuyo valor óptimo se obtiene por la minimización de (1.9), o sea

$$\mathbf{w}_{opt} = \arg\min_{\mathbf{w}} R_{emp}(\mathbf{w}) \tag{1.10}$$

Por tanto, la función $f(\mathbf{x}, \mathbf{w}_{opt})$ será aquella que proporcione la solución con el menor coste sobre el conjunto de muestras. Este principio se conoce como el de la minimización del *riesgo empírico* [Vapnik, 1995].

Evidentemente, el diseño del clasificador debe garantizar buenas propiedades de generalización, es decir, el clasificador debe conservar un buen rendimiento con datos que no han sido utilizados durante el entrenamiento. Para ello, será necesario medir las prestaciones sobre conjuntos de datos diferentes para el entrenamiento, test y, posiblemente, validación.

Existe una gran variedad de estos métodos, entre los que destacan: las Redes Neuronales Artificiales (ANNs, "Artificial Neural Networks") [Bishop, 1995, Duda et al., 2001, Haykin, 2007]; los métodos basados en núcleos [Schölkopf and Smola, 2002](Procesos Gaussianos, GPs, "Gaussian Processes" [Rasmussen and Williams, 2005], Máquinas de Vectores Soporte, SVMs, "Support Vector Machines" [Vapnik, 1998, Vapnik, 1995]); los Sistemas Expertos [Michalski, 1980] y los Árboles de Decisión [Breiman et al., 1984]; y los modelos gráficos, como, por ejemplo, las Redes Bayesianas [Buntine, 1996].

1.2.1. Redes Neuronales Artificiales

Las ANNs son redes formadas por unidades llamadas neuronas que reciben y transmiten información en forma de valores numéricos a través de interconexiones sinápticas. Normalmente las neuronas se encuentran dispuestas en capas y forman una arquitectura en paralelo, lo que dota a las ANNs de ciertas características estructurales muy ventajosas, tales como rapidez y robustez. En esta Tesis se consideran sólo las redes neuronales con conexión hacia adelante ("feed-forward"), que son demostradamente aproximadores universales [Cybenko, 1989, Hornik, 1991].

Existen dos tipos principales de redes neuronales con conexión hacia adelante: el Perceptron Multi-Capa (MLP, "Multilayer Perceptron") y las Redes de Funciones de Base Radiales (RBFN, "Radial Basis Function Network") [Bishop, 1995, Haykin, 2007], cuyo funcionamiento revisamos a continuación.

El Perceptrón Multi-Capa

Un MLP es una red formada por un conjunto de unidades sensoriales que constituyen la capa de entrada, una o más capas ocultas y una capa de salida. Esta arquitectura permite crear una relación no lineal entre un conjunto de variables de entrada y un conjunto de variables de salida. En concreto, los datos de entrada se propagan capa por capa activando nodos de cómputo intermedios compuestos por funciones de activación no lineales, hasta alcanzar la capa de salida.

En esta Tesis, se considerarán MLPs de una sola capa oculta con M neuronas y de una sola variable de salida, cuya arquitectura se muestra en la Figura 1.2, donde las funciones de activación son del tipo sigmoidal logística para ambas capas.

El modelo descrito se puede formular como:

$$f(\mathbf{x}) = \tilde{g}\left(\sum_{j=0}^{M} w_j^{(2)} g\left(\sum_{i=0}^{d} w_{ji}^{(1)} x_i\right)\right) \tag{1.11}$$

donde $\left\{w_{ji}^{(1)}\right\}$, $\left\{w_j^{(2)}\right\}$, con $i = 0, \ldots, d$ y $j = 1, \ldots, M$, son los pesos de la capa de entrada y la capa de salida, respectivamente, y $\tilde{g}$, g, las activaciones.

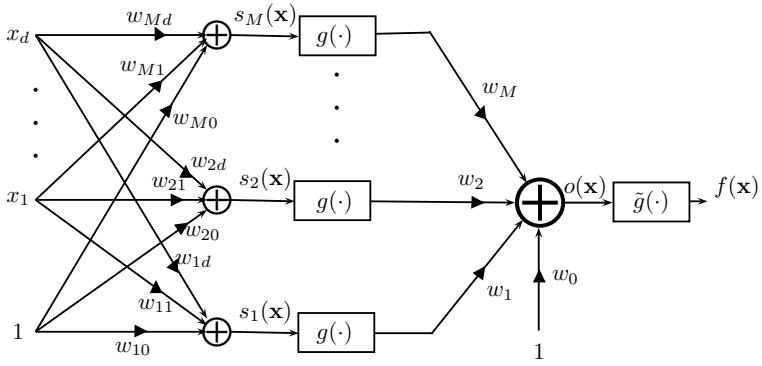


Figura 1.2: Estructura del Perceptrón Multi-Capa de una sola capa oculta y una sola variable de salida

El entrenamiento de un MLP consiste en ajustar los pesos según un conjunto de muestras etiquetadas, habitualmente mediante el algoritmo de retropropagación ("backpropagation"), o bien usando métodos de optimización más avanzados (por ejemplo, Levenberg-Marquardt, quasi-Newton, gradiente conjugado, etc.) [Bishop, 1995, Haykin, 2007], hasta que la discrepancia entre la salida deseada y la respuesta real de la red se haya minimizado. Este proceso suele presentar dificultades y debe hacerse con particular cuidado para conseguir una buena generalización; es decir, que la red entrenada sea capaz de producir salidas razonables para datos que no le han sido presentados con anterioridad.

Redes de Funciones de Base Radiales

La arquitectura de una RBFN consta de tres capas diferentes. Una capa de entrada formada por los nodos de origen, una capa oculta integrada por un conjunto de Funciones de Base Radiales (RBF, "Radial Basis Functions"), y una capa de salida que obtiene la respuesta de la red como una combinación lineal de las salidas de las RBFs. La salida de la RBFN para un dato de entrada $\mathbf{x}$ viene dada por la siguiente

expresión:

$$f(\mathbf{x}) = g\left(\sum_{n=0}^{N} w_n a_n(\mathbf{x})\right) \tag{1.12}$$

donde g es la activación, $\{w_n\}$ son los pesos de la capa de salida, y $\{a_n(\mathbf{x})\}$, las salidas de las RBFs. Supuestas éstas Gaussianas, su valor se calcula como:

$$a_n(\mathbf{x}) = \begin{cases} 1, & n = 0 \\ \exp\left(-\|\mathbf{x} - \mathbf{c}_n\|^2 / 2{\sigma_n}^2\right), & 1 \leq n \leq N \end{cases} \tag{1.13}$$

donde $\{\mathbf{c}_n\}$ son los centroides y $\{{\sigma_n}^2\}$ los parámetros de dispersión de las funciones Gaussianas.

Generalmente, el entrenamiento de una RBFN se realiza en dos etapas: en la primera etapa se aprenden las funciones base y sus respectivos parámetros. Cabe mencionar que la localización de centroides para funciones exponenciales ha sido muy estudiada por muchos autores. Por un lado, para el caso de regresión, después de la propuesta pionera de Moody y Darken [Moody and Darken, 1989], quienes sugirieron utilizar el algoritmo de K-medias, son muchas las alternativas que han aparecido. Entre ellas podemos destacar la selección secuencial basada en mínimos cuadrados ortogonales (OLS, "Orthogonal Least Squares") [Chen et al., 1991], los métodos constructivos [Platt, 1991, Fritzke, 1994], y algunos algoritmos basados en agrupamiento [Elanayar and Shin, 1994][Chen, 1995], entre otros.

En el caso de las máquinas de decisión, las cosas son diferentes porque la ubicación de la frontera de decisión es importante. En [Chang and Lippmann, 1993] y [Almeida et al., 2000] se propone seleccionar centroides entre vectores representativos de agrupamientos, [Lyhyaoui et al., 1999] presenta una serie de posibilidades después de seleccionar el primer agrupamiento, mientras [Schölkopf et al., 1997] propone utilizar como centroides los vectores soporte de una SVM entrenada.

En la segunda etapa, una vez que los centros y los parámetros de dispersión se han fijado, se aprenden los valores del vector de los pesos de la capa de salida. Normalmente, sus valores se pueden ajustar utilizando pseudo-inversión o el algoritmo de descenso por gradiente.

1.2.2. Máquinas de Vectores Soporte

Las SVMs son clasificadores basados en núcleos diseñados para proporcionar el hiperplano clasificador óptimo en un espacio de alta dimensión (denominado espacio de características, $\mathcal{H}$), generado por una función de transformación $\Phi(\cdot) : \mathcal{R}^d \rightarrow \mathcal{H}$ [Vapnik, 1998, Vapnik, 1995]. Por lo general, la solución de la SVM se expresa mediante la siguiente función discriminante:

$$f(\mathbf{x}) = \mathbf{w}^\top \Phi(\mathbf{x}) + b, \tag{1.14}$$

donde $\mathbf{w}$ y b determinan el hiperplano separador en el espacio de características. Para forzar una solución de forma que la distancia entre el hiperplano óptimo y la muestra de entrenamiento más cercana —*margen*— sea máxima, $\mathbf{w}$ y b se pueden obtener como la solución del siguiente problema de optimización [Schölkopf and Smola, 2002]:

$$\begin{array}{lll} \underset{\mathbf{w},b,\xi}{\text{mín}} & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{l=1}^{L}\xi^{(l)} & \\ \text{s.t.} & y^{(l)}\left[\mathbf{w}^\top\Phi(\mathbf{x}^{(l)}) + b\right] \geq 1 - \xi^{(l)}; & l = 1,\ldots,L \\ & \xi^{(l)} \geq 0; & l = 1,\ldots,L \end{array} \tag{1.15}$$

donde C es un parámetro positivo que controla el compromiso entre la sencillez del modelo y el error de clasificación, y $\{\xi^{(l)}\}_{l=1}^{L}$ es el conjunto de las variables de holgura que se introducen para permitir que algunas muestras estén dentro del margen o mal clasificadas (véase la Figura 1.3). El funcional de este problema de optimización es convexo, lo que garantiza la unicidad de la solución.

Para resolver (1.15), se introducen los multiplicadores de Lagrange $\{a^{(l)}\}_{l=1}^{L}$ y las correspondientes condiciones de Karush-Kuhn-Tucker (KKT) [Karush, 1939, Kuhn and Tucker, 1951], transformando el problema de optimización en el de pro-

gramación cuadrática (QP, "Quadratic Programming")

$$\begin{aligned} \max_{\mathbf{a}} \quad & -\frac{1}{2}\sum_{l=1}^{L}\sum_{l'=1}^{L} a^{(l)}a^{(l')}y^{(l)}y^{(l')}K(\mathbf{x}^{(l)},\mathbf{x}^{(l')}) + \sum_{l=1}^{L} a^{(l)} \\ \text{s.t.} \quad & \sum_{l=1}^{L} a^{(l)}y^{(l)} = 0 \\ & C \geq a^{(l)} \geq 0; \qquad l = 1,\ldots,L \end{aligned} \tag{1.16}$$

donde $K(\cdot,\cdot)$ es el núcleo asociado a $\Phi(\cdot)$. Son tres las funciones núcleo típicas, RBF (Gaussiana), polinómica y lineal; en este mismo orden,

$$\begin{aligned} K(\mathbf{x}^{(l)},\mathbf{x}^{(l')}) &= \exp\left(-\left\|\mathbf{x}^{(l)}-\mathbf{x}^{(l')}\right\|^2/2\sigma^2\right), \\ K(\mathbf{x}^{(l)},\mathbf{x}^{(l')}) &= \left(\mathbf{x}^{(l)}.\mathbf{x}^{(\mathbf{l}')}+\mathbf{1}\right)^p, \\ K(\mathbf{x}^{(l)},\mathbf{x}^{(l')}) &= \left(\mathbf{x}^{(l)}.\mathbf{x}^{(\mathbf{l}')}\right). \end{aligned}$$

El clasificador SVM no lineal toma la forma:

$$f(\mathbf{x}) = \sum_{l=1}^{L} a^{(l)}y^{(l)}K(\mathbf{x}^{(l)},\mathbf{x}) + b \tag{1.17}$$

donde $\left\{a^{(l)}\right\}$ son constantes con valores reales y positivos que se corresponden con la solución del problema QP. Una de las ventajas de esta formulación radica en que sólo aquellos datos con $a^{(l)} > 0$ forman parte de la solución. Estos puntos se denominan Vectores Soporte (SVs, "Support Vectors"), y corresponden a muestras de entrenamiento mal clasificadas o a muestras que se encuentran ubicadas en o dentro del margen .

Seguidamente, para determinar el valor de b se consideran los SVs que satisfagan la condición

$$y^{(l)}\left(\sum_{l'\in S} a^{(l')}y^{(l')}K(\mathbf{x}^{(l)},\mathbf{x}^{(l')}) + b\right) = 1 \tag{1.18}$$

donde S indica el conjunto de índices de los SVs. El valor de b se calcula como

$$b = \frac{1}{N_{S'}}\sum_{l\in S'}\left(y^{(l)} - \sum_{l'\in S} a^{(l')}y^{(l')}K(\mathbf{x}^{(l)},\mathbf{x}^{(l')})\right) \tag{1.19}$$

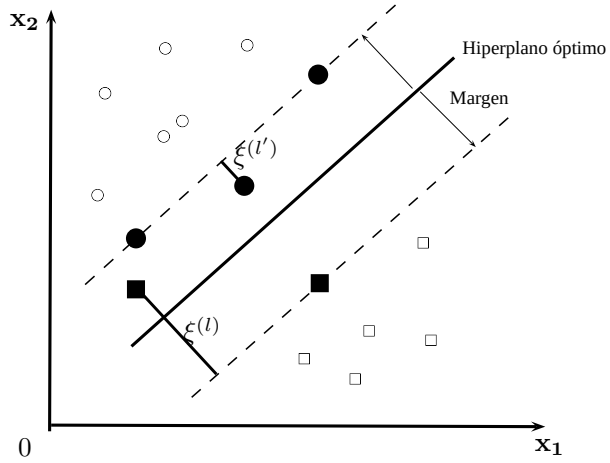


Figura 1.3: Caso no separable en un problema de dos dimensiones

donde S' indica el conjunto de índices de muestras que tienen $0 < a^{(l)} < C$.

Finalmente, el valor de los coeficientes $a^{(l)}$ define las siguientes situaciones:

1. Si $a^{(l)} = 0$, indica que $\mathbf{x}^{(l)}$ está bien clasificado y se ubica fuera del margen.

2. Cuando $0 < a^{(l)} < C$, se tiene que $\mathbf{x}^{(l)}$ está estrictamente sobre el margen y por tanto bien clasificado.

3. Si $a^{(l)} = C$, implica que $\mathbf{x}^{(l)}$ se ubica dentro del margen y puede estar mal clasificado o correctamente clasificado.

1.3. Conjuntos de máquinas de aprendizaje

Los Conjuntos de Máquinas de Aprendizaje (CMAs) han sido ampliamente utilizados en la mejora del rendimiento de una máquina única. Esta técnica tiene su origen en el trabajo de Salamon y Hansen [Hansen and Salamon, 1990], quienes demostraron que las prestaciones de una ANN pueden ser mejoradas significativamente mediante la combinación de un cierto número de ellas. Son muchos los métodos para

construir CMAs que se han propuesto [Sharkey, 1999, Kuncheva, 2004, Rokach, 2010, Schapire and Freund, 2012].

Como se puede comprobar en [Sharkey, 1999, Haykin, 2007], los CMAs pueden clasificarse de acuerdo con distintos criterios; por ejemplo, atendiendo a la manera que tienen los aprendices de ayudarse entre sí para resolver el problema. Según este criterio, los CMAs pueden dividirse en:

- **Comités**, donde todos los aprendices resuelven el mismo problema de forma independiente y, luego, se combinan sus salidas utilizando algún criterio adecuado. Para conseguir que la salida del conjunto presente prestaciones mejores que los clasificadores individuales, es necesario que la fusión combine adecuadamente las salidas, de modo que los errores cometidos en cada una de ellas sean corregidos por las otras. Para ello, las soluciones individuales deben generalizar de manera diferente, es decir, debe existir diversidad entre las soluciones.

  Para forzar diversidad entre los clasificadores que van a componer el comité, se pueden emplear distintas condiciones iniciales de entrenamiento, distintas topologías, distintos algoritmos de entrenamiento, o técnicas de reducción de dimensiones, remuestreo o filtrado para que cada clasificador emplee un conjunto de entrenamiento distinto.

  Las técnicas de remuestreo han dado lugar a la mayoría de los métodos de construir comités, destacando los que utilizan técnicas elaboradas ("Bagging", Bootstrap AGGregatING) [Breiman, 1996a], y aquéllos que emplean conjuntos de entrenamiento disjuntos ("Bootstrapping" sin reemplazamiento) [Sharkey, 1999].

  Otro aspecto importante en la construcción de comités es la adecuada combinación de las salidas de las máquinas individuales. Para ello existen múltiples opciones, tal como se describe en [Kuncheva, 2004], siendo frecuentes las combinaciones lineales (ya sea mediante un promediado directo de las salidas de todas las máquinas o empleando un conjunto de pesos que deben ser entrena-

dos), y también los métodos basados en mayorías.

- **Consorcios**, donde las máquinas que conforman el conjunto cooperan entre sí. Se pueden diferenciar dos maneras de formar consorcios: la primera, siguiendo el principio de divide y vencerás, descompone el problema a resolver en un número de subproblemas, y luego asigna a cada uno de ellos un experto diferente, como es el caso de los conjuntos modulares o Mezclas de Expertos ("Mixture of Experts", MoE) [Jacobs et al., 1991, Jordan and Jacobs, 1994]. La segunda, consiste en añadir máquinas al conjunto de modo que paso a paso se mejore la solución proporcionada por las máquinas previamente incorporadas, como es el caso de los métodos "Boosting" [Schapire and Freund, 2012].

Es importante mencionar que estas filosofías no son excluyentes entre sí: es posible la combinación entre ellas, dando lugar a otro tipo de combinaciones [Kuncheva, 2004].

A continuación, revisaremos los métodos de MoE, Bagging y Boosting.

1.3.1. Mezcla de expertos

La Mezcla de Expertos es un procedimiento para combinar aprendices (expertos) ampliamente estudiado desde su introducción en [Jacobs et al., 1991]. En una MoE, la salida del conjunto se obtiene con la ayuda de una puerta que asigna un peso de combinación convexa a la salida de cada experto en función de la localización del dato de entrada. Es decir, la puerta determina la fiabilidad de cada experto y regula su influencia en la salida de la red general. En la Figura 1.4 se muestra su estructura básica. La salida de la red es:

$$F_T(\mathbf{x}) = \sum_{t=1}^{T} g_t(\mathbf{x}) f_t(\mathbf{x}) \tag{1.20}$$

donde la salida de la puerta, $g_t(\mathbf{x})$, puede ser interpretada como la probabilidad de que la muestra $\mathbf{x}$ se atribuya al t-ésimo experto. Con el fin de asegurar esta interpretación

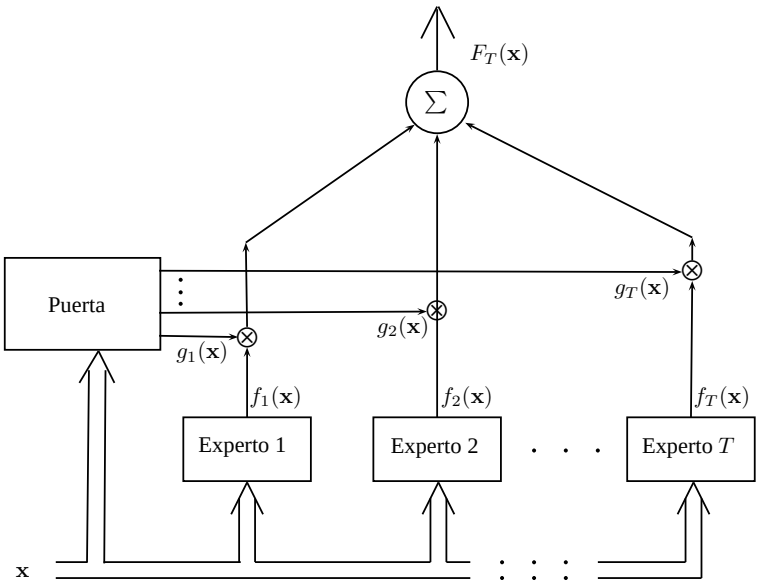


Figura 1.4: Esquema de un clasificador MoE

probabilística, la puerta incorpora a su salida una función de activación del tipo softmax [Jacobs et al., 1991].

Después del primer artículo sobre la MoE, se han desarrollado un gran número de variantes de este paradigma. En [Jordan and Jacobs, 1994], la idea original se extendió a una mezcla de modelos jerárquicos, donde se hace uso del algoritmo EM para ajustar los parámetros de la MoE [Jordan and Xu, 1995, Peng et al., 1996, Jacobs et al., 1997]. Otros de los trabajos se han enfocado al análisis teórico de la MoE [Ge and Jiang, 2006a, Ge and Jiang, 2006b] y las MoE para datos incompletos [Liao et al., 2007]. Un reciente artículo de revisión [Yuksel et al., 2012] permite conocer el estado del arte de estos diseños.

1.3.2. Bagging

La técnica de Bagging fue propuesta por Breiman [Breiman, 1996a], basándose en los métodos de "Bootstrapping" y de agregación [Dietterich, 2000]. En este

algoritmo, cada aprendiz se entrena con un conjunto de L muestras de entrenamiento, seleccionadas mediante un muestreo aleatorio con reemplazamiento del conjunto original; es decir, puede haber muestras que se repitan. Una vez entrenados los clasificadores base, la salida de la máquina conjunta se obtiene promediando o por mayoría de las salidas de todos los aprendices.

Una de las principales características de Bagging es que cuando los aprendices son máquinas inestables, tales como los árboles de decisión, presentan buenas prestaciones, pero la explicación de este comportamiento aun no está clara. En [Friedman, 1997] se sugiere que dicho comportamiento se debe a que Bagging, al combinar los aprendices, reduce la varianza y mantiene el sesgo.

1.3.3. Boosting

Boosting es uno de los métodos más potentes de aprendizaje máquina introducido en los últimos veinte años. Originalmente se diseño para la resolución de problemas de clasificación binaria y, posteriormente, se extendió a problemas de múltiples clases y regresión [Schapire and Freund, 2012]. Desde sus inicios ha despertado gran interés en la comunidad científica por el excelente rendimiento que ha demostrado al ser aplicado a diferentes tipos de problemas reales.

La idea básica de Boosting es prestar atención a las muestras difíciles de clasificar. Siguiendo esta idea, en 1990 Schapire introdujo el primer método de Boosting, para filtrado [Schapire, 1990]. Años más tarde, Freund y Schapire propusieron el primer algoritmo de Boosting que ha merecido gran atención por sus excelentes características, el algoritmo "AdaBoost" [Freund and Schapire, 1996] que combina aprendices con salidas discretas. Posteriormente, "Real AdaBoost" [Schapire and Singer, 1999] extiende esta idea para aprendices con salidas de valores reales, de tal forma que no sólo se tenga en cuenta la salida de cada componente, sino también la confianza de dichas predicciones. Desde entonces, son muchas las extensiones o modificaciones que han surgido proporcionando mejoras en sus prestaciones. En

[Meir and Rätsch, 2003, Schapire and Freund, 2012] se recogen las más relevantes. Todos estos algoritmos se caracterizan por entrenar consecutivamente clasificadores con tasas de acierto ligeramente mejores que el azar ("débiles"), empleando para ello poblaciones de muestras enfatizadas y formando el conjunto final mediante una combinación lineal de las salidas de estos clasificadores. Aunque todas estas versiones emplean los mismos principios que el AdaBoost original, presentan claras diferencias en su implementación.

La primera aplicación práctica del algoritmo AdaBoost fue en un problema de reconocimiento óptico de caracteres (OCR, "Optical Character Recognition") [Freund and Schapire, 1996]. Desde entonces, los algoritmos de Boosting se han aplicado con éxito en muchos otros problemas reales como, por ejemplo, en la clasificación de textos [Schapire and Singer, 2000], en la detección de objetos [Viola and Jones, 2001, Torralba et al., 2004, Opelt et al., 2006, Eads et al., 2009], en la clasificación de rostros [Viola and Jones, 2004], y en la recuperación de imágenes [Tieu and Viola, 2004].

Aunque AdaBoost y Real AdaBoost son los algoritmos de Boosting más conocidos, existen muchos otros algoritmos de esta clase que también han cobrado importancia. En el capítulo 2 de esta Tesis revisaremos con mayor detalle estos algoritmos.

1.4. Motivación

Si bien se han propuesto soluciones para solventar una de las limitaciones prácticas más importantes de los métodos Boosting básicos, como es el exceso de atención a muestras mal clasificadas que resultan imposibles de recuperar durante el crecimiento del conjunto [Breiman, 1999][Freund, 2001],[Rätsch et al., 1999, Rätsch et al., 2001, Rätsch and Warmuth, 2005][Lugosi and Vayatis, 2004], poca preocupación ha habido sobre otras limitaciones de importancia de estas arquitecturas. Entre dichas limitaciones podemos mencionar:

- El restringir la fusión de los aprendices a una combinación lineal, lo que se hace por la facilidad de realizar los cálculos de los correspondientes coeficientes a partir de la función de coste de estos diseños. Sin embargo, es obvio que, al ir los aprendices prestando atención a las muestras problemáticas, una fusión global no permite reducir el efecto de las muestras clasificadas de forma decididamente erróneas. Según lo dicho, cabe la cuestión: ¿por qué no reemplazar la combinación lineal de aprendices por una agregación de carácter local?. Esta idea ha sido explorada en [Meir et al., 2000], bajo una parametrización de ML por medio del algoritmo Esperanza-Maximización Generalizado (GEM, "Generalized Expectation Maximization"), utilizando como aprendices perceptrones de una sola capa oculta. Los resultados obtenidos son moderadamente buenos.

- El uso de clasificadores débiles —normalmente árboles de decisión—, lo que da como resultado estructuras complejas cuando se trata problemas de tipo local (con importantes variaciones de las fronteras en entornos pequeños). Esto se debe a la poca capacidad expresiva de los aprendices. El algoritmo Real AdaBoost, cuando no consigue clasificar algunas muestras problemáticas, continúa enfatizando iteración tras iteración y añadiendo aprendices hasta conseguir clasificarlas correctamente; por tanto, proporciona soluciones de arquitecturas de gran tamaño y sobreajustadas, con poca capacidad de generalización. ¿Por qué no utilizar clasificadores con mayor capacitad expresiva como aprendices de Boosting?. En el caso de aprendices globales, existen trabajos donde se han utilizado clasificadores considerados como fuertes [Gómez-Verdejo et al., 2006, Gómez-Verdejo et al., 2008, Schwenk and Bengio, 2000, Rätsch et al., 2001, Mayhua-López et al., 2010, Mayhua-López et al., 2012] consiguiendo estructuras más compactas (de menor número de aprendices) con prestaciones mejores.

  Parece, pues, que si se empleasen estructuras de núcleos tipo SVMs [Schölkopf and Smola, 2002, Shawe-Taylor and Cristianini, 2004, Vapnik, 1995] como aprendices, se podrían conseguir mayores ventajas.

Sin embargo, los trabajos en esta línea no parecen mostrar los resultados esperados [Wickramaratna et al., 2001, Kim et al., 2003, Rangel et al., 2005, Li et al., 2008, Wu et al., 2009, Lima et al., 2009, Wei et al., 2010]; lo que se debe a que el uso de SVMs como elementos base no es sencillo, ya que, debido a la falta de diversidad entre los aprendices —que son estables—, su adición directa al conjunto tiende a degradar el rendimiento global [Wickramaratna et al., 2001].

A la vista de estas limitaciones de los algoritmos de Boosting, parece muy recomendable la utilización de elementos locales —sea como elemento de fusión, sea como aprendiz—, con el fin de mejorar las prestaciones de esta familia de CMAs.

1.5. Objetivos de la Tesis

El objetivo principal perseguido en esta Tesis consiste en reducir las limitaciones anteriormente reseñadas:

- En primer lugar, tomando como base el esquema de fusión que caracteriza a las MoEs, se introducen puertas locales sencillas —con estructuras de núcleos— para mejorar el esquema de fusión de Boosting sin alterar el modo de diseño de los aprendices, pero entrenando las puertas locales para que la fusión minimice (paso a paso) la función de coste propia de estos esquemas.

- En segundo lugar, se emplean estructuras de núcleos como aprendices. Para ello, se ha diseñado un procedimiento adecuado que permite controlar la complejidad y diversidad en los clasificadores tipo SVM, de modo tal que su uso como aprendiz de Boosting es viable, consiguiendo estructuras más compactas y con mejores prestaciones que la original.

Estas modificaciones se han abordado en problemas de clasificación binaria sobre bases de datos con diferentes características (tamaño, dimensión y dificultad),

obteniendo los resultados esperados.

1.6. Estructura del documento

El resto de los capítulos de esta Tesis se organiza según se indica a continuación: en el **Capítulo 2**, se revisan los fundamentos de Boosting, incluyendo un comentario básico del algoritmo AdaBoost y con más detalle el Real AdaBoost. En el **Capítulo 3**, se presenta el algoritmo de fusión controlada por puerta, detallando sus fundamentos, junto con su evaluación experimental. Seguidamente, en el **Capítulo 4** se explica un nuevo procedimiento para entrenar SVMs como aprendices de RAB, que también se evalúan del modo habitual. Finalmente, en el **Capítulo 5**, se hace una reflexión sobre el trabajo realizado y su relevancia, así como sobre las líneas de investigación que siguen abiertas y que sería interesante abordar en un futuro próximo.

Capítulo 2

Boosting y el algoritmo Real AdaBoost

Los métodos de Boosting son un grupo de algoritmos ampliamente estudiados. En este capítulo se recordarán sus fundamentos, incluyendo un comentario básico del algoritmo AdaBoost (AB). A continuación, se analiza en detalle la primera extensión del algoritmo AB para clasificadores base con un rango de salida continuo y no limitado al conjunto de valores $\{-1, 1\}$, conocida como "Real AdaBoost" (RAB). Seguidamente, se revisan las variantes que maximizan de forma directa el margen de clasificación[1], las versiones para datos ruidosos y, finalmente, aquéllos con características de interés en el contexto de esta Tesis.

2.1. Fundamentos de Boosting

El algoritmo Boosting tienen su origen en la llamada teoría PAC ("Probably Approximately Correct") propuesta por Valiant [Valiant, 1984]. Kearns y Valiant, trabajando dentro de este marco, se preguntaron si se pueden combinar clasifica-

[1]En este capítulo, el valor del margen de una muestra $\mathbf{x}$ con etiqueta y con respecto al clasificador $f(\mathbf{x})$ se define como $yf(\mathbf{x})$.

dores "débiles" para dar lugar a un clasificador con prestaciones arbitrariamente elevadas, "fuerte" [Kearns and Valiant, 1994]. Fue Schapire quien introdujo el primer algoritmo de Boosting en forma de filtrado [Schapire, 1990], y posteriormente Freund propuso una forma simple y elegante de combinar linealmente clasificadores débiles [Freund, 1995]. Años más tarde, Freund y Schapire introdujeron el primer algoritmo Boosting entrenable, denominado "AdaBoost" ("ADAptive Boosting") [Freund and Schapire, 1996]. Desde entonces son muchos los algoritmos Boosting que se han propuesto. Una descripción más detallada del origen de los algoritmos Boosting se puede encontrar en [Schapire and Freund, 2012].

AB es un algoritmo práctico y eficiente que trabaja aplicando una distribución de pesos o de énfasis sobre las muestras. En cada ronda, esta distribución se usa para entrenar un clasificador débil de tal manera que preste mayor atención a las muestras que son más difíciles de clasificar. Seguidamente, a la salida del clasificador débil se le asigna un peso que depende del error de entrenamiento ponderado que comete ese clasificador. Luego, la función de énfasis se actualiza de forma que se asigna más peso a las muestras que son difíciles de clasificar y menos peso a las muestras que son fáciles de clasificar. Finalmente, la salida del conjunto clasificador se obtiene como una combinación lineal ponderada de todos los clasificadores base.

Una de las limitaciones del algoritmo AB es que combina clasificadores cuyas salidas son discretas. "Real AdaBoost" (RAB) [Schapire and Singer, 1999] extiende esta idea para clasificadores base cuyas salidas son valores reales, de tal forma que no sólo se tenga en cuenta el criterio de cada componente, sino también la "confianza" de dichas predicciones.

En la siguiente sección se analiza en detalle el algoritmo RAB, describiendo, en primer lugar, su funcionamiento, y discutiendo a continuación sus principales características.

2.2. El algoritmo Real AdaBoost

Dado un conjunto de muestras correctamente etiquetadas correspondiente a un problema de clasificación binaria, $\left\{\mathbf{x}^{(l)}, y^{(l)}\right\}_{l=1}^{L}$, donde $\mathbf{x}^{(l)} \in \mathcal{R}^d$ es una muestra de dimensión d e $y^{(l)} \in \{-1, +1\}$ su correspondiente etiqueta, el objetivo del algoritmo RAB es construir una función capaz de clasificar nuevas muestras lo más correctamente posible.

El algoritmo RAB entrena una serie de clasificadores con prestaciones ligeramente mejores que el azar (débiles), llamados clasificadores base, y los combina de modo que se obtenga un clasificador con elevadas prestaciones (fuerte). Para ello, durante una serie de iteraciones o rondas, $t = 1, ..., T$, entrena un clasificador que implementa una función $f_t(\mathbf{x}) \in [-1, 1]$, asignándole un peso de salida, α_t, y lo añade al conjunto, de modo que la salida global del sistema, $F_T(\mathbf{x})$, se obtiene como combinación lineal ponderada de todos los clasificadores base (véase la Figura 2.1)

$$F_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x}). \tag{2.1}$$

La etiqueta asignada por el sistema a un nuevo dato viene dada por

$$\hat{y}(\mathbf{x}) = \text{sign}\left[F_T(\mathbf{x})\right], \tag{2.2}$$

2.2.1. Entrenamiento de los clasificadores base y énfasis

El clasificador base generado en la t-ésima ronda se entrena para minimizar el error cuadrático medio enfatizado del conjunto de datos original, es decir, para que minimice

$$C_t = \sum_{l=1}^{L} D_t(\mathbf{x}^{(l)})[y^{(l)} - f_t(\mathbf{x}^{(l)})]^2 \tag{2.3}$$

donde $D_t(\cdot)$ es la función de énfasis que indica la importancia que el clasificador debe asignar a cada muestra. Inicialmente se otorga la misma importancia a todas

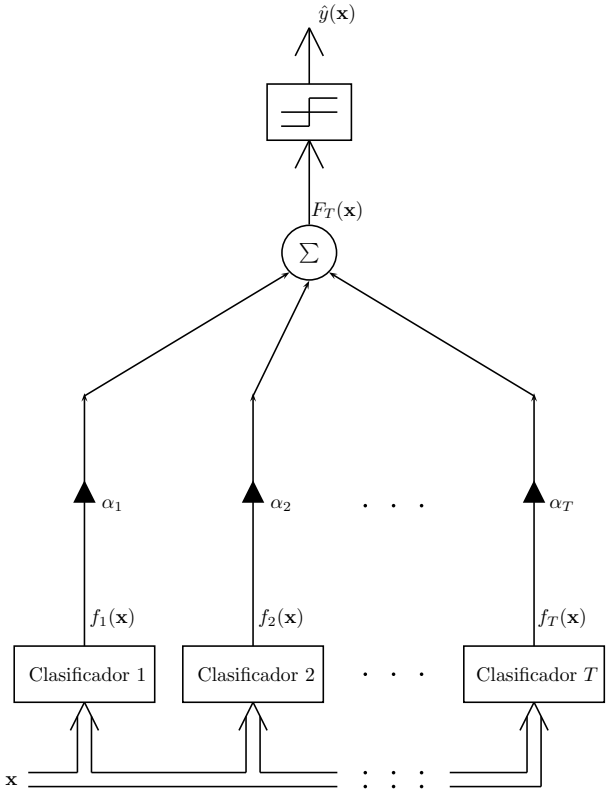


Figura 2.1: Esquema de un clasificador RAB.

las muestras, es decir,

$$D_1(\mathbf{x}^{(l)}) = 1/L, \quad l = 1, ..., L,$$

y en cada ronda esta función se actualiza para que el siguiente clasificador preste mayor atención a los patrones que han sido clasificados erróneamente por los clasificadores anteriores. Para ello, se emplea la regla de actualización

$$D_{t+1}(\mathbf{x}^{(l)}) = \frac{D_t(\mathbf{x}^{(l)}) \exp[-\alpha_t f_t(\mathbf{x}^{(l)}) y^{(l)}]}{Z_t}, \tag{2.4}$$

donde Z_t es un factor de normalización, que se calcula mediante

$$Z_t = \sum_{l=1}^{L} D_t(\mathbf{x}^{(l)}) \exp[-\alpha_t f_t(\mathbf{x}^{(l)}) y^{(l)}] \tag{2.5}$$

para asegurar que $\sum_{l=1}^{L} D_{t+1}(\mathbf{x}^{(l)}) = 1$.

2.2.2. Cálculo de los pesos de salida

El cálculo del peso de salida asociado al clasificador t-ésimo, α_t, se realiza minimizando en cada iteración la función de coste:

$$B_t = \frac{1}{L}\sum_{l=1}^{L}\exp[-F_t(\mathbf{x}^{(l)})y^{(l)}], \tag{2.6}$$

donde F_t es la salida (parcial) de la máquina conjunta correspondiente a la t-ésima ronda:

$$F_t(\mathbf{x}) = \sum_{t'=1}^{t}\alpha_{t'}f_{t'}(\mathbf{x}) \tag{2.7}$$

El hecho de usar esta función de coste, B_t, se debe a que es una cota superior del error de entrenamiento, es decir

$$E_t = \frac{1}{2L}\sum_{l=1}^{L} \mid \mathrm{sign}[F_t(\mathbf{x}^{(l)})] - y^{(l)} \mid \leq B_t \tag{2.8}$$

Tal y como puede verse en [Schapire and Singer, 1999], el procedimiento seleccionado para la minimización de B_t conduce a distintos métodos para el cálculo de los $\{\alpha_t\}_{t=1}^{T}$, dando lugar a distintas implementaciones del algoritmo. Entre este conjunto de posibilidades se ha escogido el método que emplea el AB original, válido para el RAB cuando el rango de valores de salida de los clasificadores se encuentra en el intervalo $[-1, 1]$. En este caso, en lugar de minimizar directamente B_t, se recurre a la minimización de la siguiente cota superior[2]:

$$B_t = \frac{1}{L}\sum_{l=1}^{L}\exp[-F_{t-1}(\mathbf{x}^{(l)})y^{(l)}]\exp[-\alpha_t f_t(\mathbf{x}^{(l)})y^{(l)}] \leq$$

[2]La demostración de la existencia de dicha cota se obtiene probando que la función $f(x) = \exp[-\alpha_t x] - \frac{1+x}{2}\exp(-\alpha_t) - \frac{1-x}{2}\exp(\alpha_t)$ es positiva en el intervalo $[-1, 1]$, lo que se logra viendo que $f(x)$ se anula en $x = \pm 1$ y que su derivada segunda, $f''(x)$, es positiva $\forall x$. A partir de este resultado, es inmediato establecer que $\exp[-\alpha_t f_t(\mathbf{x}^{(l)})y^{(l)}] \leq \frac{1+f_t(\mathbf{x}^{(l)})y^{(l)}}{2}\exp(-\alpha_t) + \frac{1-f_t(\mathbf{x}^{(l)})y^{(l)}}{2}\exp(\alpha_t)$ para $f_t(\mathbf{x}^{(l)})y^{(l)} \in [-1, 1]$, de donde se deduce fácilmente (2.9).

$$\leq \frac{1}{L}\sum_{l=1}^{L}\exp[-F_{t-1}(\mathbf{x}^{(l)})y^{(l)}]\left[\frac{1+f_t(\mathbf{x}^{(l)})y^{(l)}}{2}\exp(-\alpha_t)+\frac{1-f_t(\mathbf{x}^{(l)})y^{(l)}}{2}\exp(\alpha_t)\right] \quad (2.9)$$

Nótese que cuando el rango de salida de los clasificadores está limitado a los valores $\{-1,1\}$, es decir, cuando se trata de la versión original del AB, se verifica la igualdad en (2.9).

Para minimizar la cota sobre B_t, se deriva la expresión anterior con respecto a α_t y se iguala a 0. De este modo, y tras algunas manipulaciones elementales, puede obtenerse la siguiente expresión para el cálculo de α_t

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1+\gamma_t}{1-\gamma_t}\right). \quad (2.10)$$

donde γ_t, denominado "edge" o parámetro de separación del t-ésimo clasificador, viene dado por

$$\gamma_t = \sum_{l=1}^{L} D_t(\mathbf{x}^{(l)})f_t(\mathbf{x}^{(l)})y^{(l)} \quad (2.11)$$

e indica la calidad de cada uno de los clasificadores, ya que mide, sobre el conjunto de entrenamiento, la correlación existente entre las salidas y las correspondientes etiquetas enfatizando la población de acuerdo con $D_t(\cdot)$.

Los detalles del desarrollo matemático que conducen a (2.10) y (2.11) se encuentran recogidos en el Apéndice A.1.

El resto de posibilidades propuestas en [Schapire and Singer, 1999] para el cálculo del conjunto de pesos $\{\alpha_t\}_{t=1}^{T}$ consisten en minimizar otras cotas de B_t, o bien minimizar directamente B_t mediante métodos numéricos. La elección de (2.10) se ha realizado tras comprobar que el empleo de un criterio u otro no produce diferencias apreciables en el comportamiento global del conjunto, es decir, las prestaciones finales son similares. De esta forma, disponer de una expresión analítica para $\{\alpha_t\}_{t=1}^{T}$ simplifica el diseño y minimiza el coste computacional sin afectar a las prestaciones obtenidas.

En la Tabla 2.1 se recoge el pseudocódigo que describe el funcionamiento de este algoritmo.

Tabla 2.1: Pseudocódigo del algoritmo RAB.

1 - Entradas: $\{\mathbf{x}^{(l)}, y^{(l)}\}_{l=1}^{L}$.

2 - Inicialización: $D_1(\mathbf{x}^{(l)}) = 1/L \quad \forall l$.

3 - Para $t = 1, \ldots, T$

3.1 - Entrenar un clasificador, $f_t(\mathbf{x}^{(l)})$, minimizando la función de coste:

$$C_t = \sum_{l=1}^{L} D_t(\mathbf{x}^{(l)})[y^{(l)} - f_t(\mathbf{x}^{(l)})]^2.$$

3.2 - Calcular el peso de salida asociado a este clasificador como

$$\alpha_t = \tfrac{1}{2} \ln\left(\tfrac{1+\gamma_t}{1-\gamma_t}\right)$$

donde $\gamma_t = \sum_{l=1}^{L} D_t(\mathbf{x}^{(l)}) f_t(\mathbf{x}^{(l)}) y^{(l)}$.

3.3 - Actualizar la función de énfasis para la siguiente iteración:

$$D_{t+1}(\mathbf{x}^{(l)}) = \tfrac{D_t(\mathbf{x}^{(l)}) \exp[-\alpha_t f_t(\mathbf{x}^{(l)}) y^{(l)}]}{Z_t}$$

siendo $Z_t = \sum_{l=1}^{L} D_t(\mathbf{x}^{(l)}) \exp[-\alpha_t f_t(\mathbf{x}^{(l)}) y^{(l)}]$.

4 - El clasificador final implementa la función:

$$F_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t f_t(\mathbf{x})$$

siendo, por tanto, $\hat{y}(\mathbf{x}) = \text{sign}\,[F_T(\mathbf{x})]$ la clase estimada para el patrón $\mathbf{x}$.

2.2.3. Propiedades del RAB

Desde la introducción del algoritmo AB, son muchos los autores que han analizado su comportamiento para justificar su buen funcionamiento. La mayoría de ellos concuerdan en dos consideraciones. La primera de ellas en la plausible rápida reducción del error de entrenamiento según crece el conjunto, mientras que la segunda en el hecho de que, aunque el error de entrenamiento llegue a hacerse cero, el error de

test o de generalización puede continuar decreciendo según se añaden más clasificadores al conjunto. En esta sección se analizan estas dos propiedades para el caso del algoritmo RAB.

Convergencia a cero del error de entrenamiento

En 1997, Freund y Schapire (en [Freund and Schapire, 1997]) demostraron que en el caso del algoritmo AB el error de entrenamiento se reduce a cero a un ritmo exponencial con el número de iteraciones. Posteriormente, al proponer la versión del RAB, extendieron este resultado a dicho algoritmo [Schapire and Singer, 1999].

La demostración de este resultado para el algoritmo RAB se realiza partiendo de la cota sobre E_t dada por (2.6) y (2.10) y considerando, además, que los valores de α_t se eligen según las ecuaciones (2.10)-(2.11). Bajo estas premisas, se puede obtener la siguiente cota del error de entrenamiento en la iteración t-ésima:

$$E_t = \frac{1}{2L}\sum_{l=1}^{L} \mid \text{sign}\left[F_t(\mathbf{x}^{(l)})\right] - y^{(l)} \mid \leq \prod_{t'=1}^{t}\sqrt{1-\gamma_{t'}^2} \tag{2.12}$$

La demostración de este resultado se presenta en el Apéndice A.2.

Teniendo ahora en cuenta que $1 - x \leq \exp(-x)$, para $x > 0$, y definiendo $\gamma^2 = \text{mín}_{t'=1,\ldots t}\{\gamma_{t'}^2\}$, (2.12) puede reescribirse en la forma

$$E_t \leq B_t \leq \prod_{t'=1}^{t}\sqrt{1-\gamma_{t'}^2} \leq \exp\left[-\frac{1}{2}\sum_{t'=1}^{t}\gamma_t^2\right] \leq \exp\left[-\frac{t}{2}\gamma^2\right] \tag{2.13}$$

Aunque el valor de γ puede variar en cada iteración, (2.13) sugiere una disminución exponencial del error de entrenamiento según aumenta el número de clasificadores en el conjunto. De hecho, esta característica ha sido empleada frecuentemente como argumento para justificar las buenas prestaciones de este algoritmo.

Análisis del error de generalización

Reducir el error de entrenamiento mediante la incorporación de nuevos clasificadores al conjunto no implica que se esté realizando un diseño sobreajustado. En la

práctica, ocurre todo lo contrario, ya que, simultáneamente a la reducción del error de entrenamiento, se está mejorando el error de generalización, incluso cuando el error de entrenamiento se ha anulado, como se desprende de [Schapire et al., 1998], que justifica su tesis en la combinación de los siguientes resultados:

1. Un mayor margen de clasificación mejora el error de generalización del conjunto. Para su explicación, téngase en cuenta que la capacidad de generalización de una red que implementa una función $F_t(\cdot)$ sobre un conjunto de datos generado de forma independiente a partir de una función de densidad de probabilidad $p(x)$ puede formularse en términos del riesgo esperado

$$R[F_t] = E_p\left[\frac{1}{2} \mid \text{sign}\left[F_t(\mathbf{x})\right] - y \mid\right], \tag{2.14}$$

siendo $E_p[\cdot]$ el operador esperanza matemática sobre la función de densidad de probabilidad $p(x)$. Además, tal y como se demuestra en [Schapire and Singer, 1999] para el caso del RAB, $R[F_t]$ se mantiene con una probabilidad de al menos $1-\beta$ ($\beta > 0$) por debajo de la cota:

$$R[F_t] \leq R_t^{\text{margin}}(\theta) + O\left(\frac{1}{\sqrt{L}}\sqrt{\frac{\vartheta \log^2 L/\vartheta}{\theta^2} + \log\frac{1}{\beta}}\right), \tag{2.15}$$

donde $O(\cdot)$ indica que el término que representa es proporcional a su argumento, L es el tamaño del conjunto de entrenamiento, ϑ es la dimensión VC del espacio de funciones en el que se encuentran los clasificadores [Vapnik and Chervonenkis, 1971], y $R_t^{\text{margin}}(\theta)$ es la fracción de datos de entrenamiento que presentan un margen de clasificación (separación entre la muestra y la frontera de clasificación) menor o igual que $\theta \in (0, 1]$; i.e.,

$$R_t^{\text{margin}}(\theta) = \frac{1}{L}\sum_{l=1}^{L} I\left\{\rho_t(\mathbf{x}^{(l)}) \leq \theta\right\}, \tag{2.16}$$

siendo $I\{\cdot\} = 1$ cuando se cumple la condición entre llaves, y 0 en el caso contrario, y donde se ha definido el margen de clasificación de cada muestra ,

$\rho_t(\mathbf{x}^{(l)})$, como

$$\rho_t(\mathbf{x}^{(l)}) = \frac{f_t(\mathbf{x}^{(l)})y^{(l)}}{\sum_{t'=1}^{t}\alpha_{t'}}, \tag{2.17}$$

de modo que $\rho_t(\mathbf{x}^{(l)})$ está comprendido entre -1 y 1, y para las muestras correctamente clasificadas su valor es tanto más próximo a 1 cuanto mayor es $f_t(\mathbf{x}^{(l)})$ (i.e., conforme mayor es la distancia a la frontera), y de forma similar es próximo a -1 para las muestras mal clasificadas.

La expresión (2.15) permite utilizar el riesgo marginal $R_t^{\text{margin}}(\theta)$ en vez de $R[F_t]$ para analizar la capacidad de generalización del conjunto, ya que indica que para reducir $R[F_t]$ lo importante es conseguir un valor pequeño de $R_t^{\text{margin}}(\theta)$.

2. El segundo resultado indica cómo se comporta el riesgo marginal $R_t^{\text{margin}}(\theta)$, mostrando que el RAB no sólo reduce el error de entrenamiento en cada iteración, sino que además es capaz de reducir $R_t^{\text{margin}}(\theta)$. Este resultado se basa en la existencia de la siguiente cota sobre el riesgo marginal[3]

$$R_t^{\text{margin}}(\theta) \leq \prod_{t'=1}^{t}(1+\gamma'_t)^{\frac{1+\theta}{2}}(1-\gamma'_t)^{\frac{1-\theta}{2}}. \tag{2.18}$$

Observando la forma de esta cota, se puede afirmar que si los factores que la componen son menores que 1, su valor irá decreciendo según se incorporen clasificadores al conjunto y, consecuentemente, el riesgo marginal, $R_T^{\text{margin}}(\theta)$, tenderá a decrecer.

No obstante, no puede asegurarse que los factores que componen dicha cota van a ser menores que 1, ya que su valor depende tanto de θ como del valor del parámetro de separación de cada clasificador, γ_t. Sin embargo, sí que es plausible que se verifique dicha condición cuando se consideran valores pequeños de θ, tal y como puede verse en la Figura 2.2, donde se representa, para distintos valores de θ, el valor del factor t-ésimo en función de γ_t. De hecho, cuando θ tiende a 0 se puede afirmar que los factores son menores que 1, y según se

[3]Véase la demostración en [Meir and Rätsch, 2003].

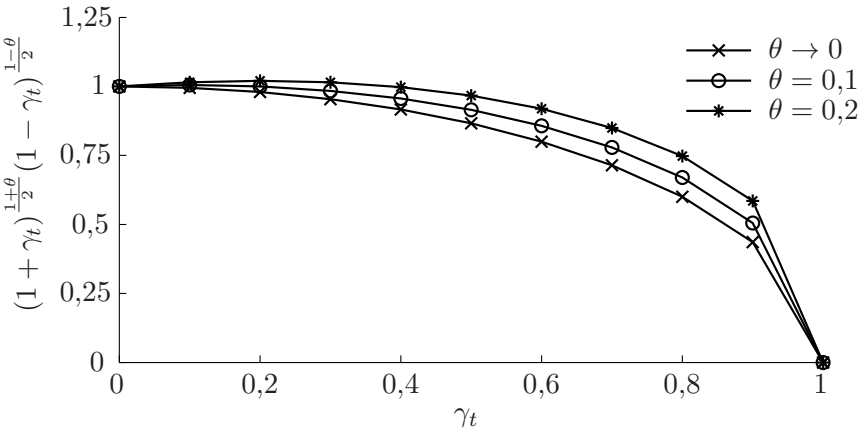


Figura 2.2: Evolución del término t-ésimo de la cota sobre el riesgo marginal en función del parámetro de separación, γ_t, para distintos valores de θ.

consideran valores más elevados de θ, se dificulta que se cumpla tal condición, dependiendo en última instancia del valor del parámetro de separación γ_t que presente el clasificador t-ésimo.

Combinando estos dos resultados, se ve cómo en cada iteración el RAB es capaz de disminuir el riesgo marginal del conjunto y, consecuentemente, reducir el error de generalización; proceso que puede continuar según se añaden clasificadores al conjunto, incluso cuando el error de entrenamiento es nulo.

2.3. Otros algoritmos de Boosting

En [Friedman et al., 2000, Friedman, 2002] se plantea el algoritmo AB como un algoritmo de regresión logística aditivo que utiliza el método de Newton para optimizar una función de coste exponencial del margen. Siguiendo este punto de vista, en [Mason et al., 2000a, Mason et al., 2000b] se plantea una generalización de los algoritmos Boosting de tal forma que permite el uso de otro tipo de funcio-

nes de coste, dando lugar a algoritmos más efectivos. Por supuesto, no todas las funciones posibles pueden ser funciones de coste: en [Duffy and Helmbold, 1999] se establecen las condiciones necesarias y suficientes para que una función pueda ser utilizada como función de coste en los algoritmos Boosting. Otras extensiones permiten el uso de funciones de coste asimétricas, para manejar los casos donde en más costoso predecir una clase que la otra [Masnadi-Shirazi and Vasconcelos, 2007, Masnadi-Shirazi and Vasconcelos, 2011].

2.3.1. Boosting y la maximización del margen

Una característica singular del algoritmo AB/RAB es su resistencia al sobreajuste, siendo capaz de seguir añadiendo clasificadores base al conjunto y mejorando las prestaciones sobre el conjunto de test, incluso cuando el error de entrenamiento se ha hecho nulo. Este comportamiento se ha analizado por varios autores [Drucker et al., 1993, LeCun et al., 1995, Drucker and Cortes, 1996, Schwenk and Bengio, 1997, Breiman, 2000, Mease and Wyner, 2008], y algunos lo han atribuido a la relación de los algoritmos de Boosting con el Máximo Margen [Schapire and Singer, 1999][Schapire et al., 1998], mientras que otros lo creen debido a la diversidad inducida por las sucesivas reponderaciones [Breiman, 1998, Breiman, 1999].

Aunque AB/RAB, de forma directa, no maximiza el margen, existen muchos algoritmos Boosting que sí lo hacen. Breiman propuso el algoritmo Arc-gv [Breiman, 1998, Breiman, 1999], que maximiza el margen, pero sólo de forma asintótica. El primer algoritmo que maximiza de forma directa el margen es el AdaBoost* [Rätsch and Warmuth, 2005]. Desde entonces se han introducido una serie de propuestas que maximizan, de forma comprobada, tanto el margen como el margen suave; entre ellos podemos mencionar el TotalBoost [Warmuth et al., 2006], el SoftBoost [Warmuth et al., 2008a], y varios otros [Rätsch et al., 2001, Rudin et al., 2004, Rudin et al., 2007].

2.3.2. Algoritmos Boosting para datos ruidosos

Dietterich [Dietterich, 2000] fue el primero en observar que las prestaciones de AB/RAB se deterioran notablemente en presencia de datos ruidosos. La razón es que la actualización exponencial de los pesos que AB/RAB asigna a cada muestra se concentra demasiado en las muestras que son las más difíciles de clasificar. Cuando algunas muestras son ruidosas, la función de énfasis tiende a concentrarse en ellas, ocasionando que dominen el proceso de entrenamiento.

Un procedimiento popular para tratar con muestras ruidosas es limitar el valor de los pesos que se les asigna, enfoque que tiene sus orígenes en las SVMs [Vapnik, 1998, Vapnik, 1995]. En la literatura sobre Boosting, este procedimiento también se conoce como "suavizar" la distribución de las muestras ("smoothing"), siendo DualLPBoost [Grove and Schuurmans, 1998] el primer algoritmo con este enfoque. También se deben citar los trabajos desarrollados por Rätsch [Rätsch et al., 2001], que presenta dos variantes de AB, el LP-AdaBoost y QP-AdaBoost.

Otra vía diferente que consigue hacer que los algoritmos de Boosting sean robustos al ruido consiste en sustituir la función de coste exponencial de AB/RAB por una función de coste que penalice las muestras mal clasificadas de forma más suave. Un ejemplo de este tipo de funciones de coste es la log–verosimilitud negativa, también conocida como función de coste logística. Ejemplos de algoritmos que siguen este camino son LogLossBoost [Collins et al., 2002] y FilterBoost [Bradley and Schapire, 2007]. Ambas visiones no son excluyentes entre sí, y existen algoritmos que optimizan la función de coste logística y al mismo tiempo suavizan la distribución de las muestras, como el ERLPBoost [Warmuth et al., 2008b].

Otros algoritmos que buscan controlar la excesiva atención que Boosting presta a las muestras ruidosas son los presentados en [Gómez-Verdejo et al., 2006][Gómez-Verdejo et al., 2008], donde se propone un método de énfasis que pondera las muestras según su error y su cercanía a la frontera, con el fin de mejorar las prestaciones del diseño estándar de RAB. Dicha

modificación del algoritmo RAB ha demostrado proporcionar buenas prestaciones sobre bases de datos de problemas reales.

2.3.3. Otros algoritmos Boosting con características de interés

Existen muchos otros algoritmos Boosting que son modificaciones de AB y RAB. Entre ellos, mencionaremos los algoritmos de aprovechamiento máximo ("leveraging"), caracterizados por emplear como función de énfasis la derivada de la función de coste aplicada para seleccionar los pesos de salida de cada clasificador [Rätsch et al., 2002b][Meir and Rätsch, 2003]; los algoritmos que optimizan de forma directa el margen, como el DOOM ("Direct Optimization Of Margin") [Mason et al., 2000a], el AnyBoost o el MarginBoost [Mason et al., 2000b]. También podemos encontrar formulaciones para dar cabida a clasificadores lineales como máquinas base [Mannor and Meir, 2002, Lu et al., 2006]; métodos que hacen uso de la regresión logística [Friedman et al., 2000] y que utilizan el procedimiento de la búsqueda estocástica [Friedman, 2002]; los algoritmos para costes sensibles al caso [Masnadi-Shirazi and Vasconcelos, 2011]; e incluso algoritmos que resultan de considerar una formulación dual del problema [Shen and Li, 2010].

Por otro lado, existen algoritmos de Boosting para tratar problemas con datos desequilibrados [Chen et al., 2010] o con datos parcialmente etiquetados (aprendizaje semisupervisado) [Mallapragada et al., 2009, Chen and Wang, 2011]. Otro de los problemas para el que han sido utilizados los algoritmos de Boosting es la selección de núcleos para construir máquinas de núcleos dispersos [Chen et al., 2006, Sun and Yao, 2010].

Por último, existen trabajos recientes que combinan métodos de Boosting con otras técnicas, como con proyecciones no lineales [García-Pedrajas et al., 2007] o con bosques de rotación [Zhang and Zhang, 2008].

2.4. Resumen

En este capítulo se han descrito, en primer lugar, los fundamentos de Boosting con un breve comentario del algoritmo AB. Seguidamente, se aborda el funcionamiento del algoritmo RAB, explicando el procedimiento a seguir para entrenar los clasificadores que lo componen, la actualización de la función de énfasis, $D_t(\cdot)$, que permite que los nuevos clasificadores presten mayor atención a las muestras más erróneas, y la manera de calcular los pesos de la capa de salida $\{\alpha_t\}_{t=1}^{T}$ que van asociados a cada clasificador.

A continuación, se han revisado las propiedades más relevantes de RAB, explicando así su comportamiento. Por un lado, se ha mostrado cómo el error de entrenamiento se reduce a un ritmo aproximadamente exponencial; y, por otro lado, se ha analizado la relación existente entre el error de generalización y el margen de clasificación, mostrando cómo el RAB es capaz de mejorar la capacidad de generalización del conjunto mediante una reducción del riesgo marginal.

Finalmente, se han revisado las principales modificaciones y variantes de los algoritmos Boosting que han aparecido en la literatura, según distintos principios y diferentes propósitos.

Capítulo 3

Boosting con Fusión Controlada por Puerta

Tal como se señaló en el capítulo anterior, el algoritmo RAB calcula su salida como una combinación lineal ponderada de las salidas de los clasificadores base que lo componen. Emplear una combinación lineal puede limitar las prestaciones del conjunto cuando se utilizan como clasificadores base máquinas globales —tales como los árboles de decisión o simples MLPs—, ya que la arquitectura de la máquina resultante también es global.

Por el contrario, la técnica de fusión de la Mezcla de Expertos (MoE) incluye una puerta entrenable, es decir, una red que según el dato presente a la entrada asigna un vector de pesos específico para la combinación de las redes que componen el conjunto. Este esquema de fusión más potente y sofisticado dota al conjunto resultante de una deseable capacidad expresiva local. Por consiguiente, es tentador tratar de mejorar las prestaciones de los algoritmos Boosting mediante la sustitución de su combinación lineal por una puerta entrenable.

Siguiendo esta idea, en este capítulo se presenta un nuevo algoritmo Boosting que incluye en su esquema de fusión una puerta entrenable. Este nuevo método, que denominaremos Real AdaBoost con Fusión Controlada por Puerta (GCF-RAB, "Gate Controlled Fusion RAB") [Mayhua-López et al., 2010, Mayhua-López et al., 2012], toma como punto de partida el algoritmo RAB, manteniendo en todo momento aquellas características que permiten conservar sus buenas propiedades y, tal como se verá en los resultados, otorgándole mejores prestaciones en términos de capacidad de generalización y, en muchos casos, de sencillez del conjunto resultante.

3.1. Estructura básica

La Figura 3.1 muestra la arquitectura de este algoritmo, que se diseña de manera similar al RAB, empezando por entrenar cada aprendiz del modo ya visto en el Apartado 2.2.

Una vez entrenado el t-ésimo clasificador base, el peso asignado a su salida es la salida de una puerta

$$\alpha_t(\mathbf{x}) = \mathbf{w}_t^\top \mathbf{a}(\mathbf{x}), \tag{3.1}$$

donde $\mathbf{w}_t$ es el vector de pesos de la capa de salida de la puerta y $\mathbf{a}(\mathbf{x}) = [a_0(\mathbf{x}),\ a_1(\mathbf{x}),\ \ldots\ , a_N(\mathbf{x})]^\top$ es el vector de salidas de las funciones de base radiales. En el caso particular en que estas funciones sean núcleos Gaussianos, tendremos que:

$$a_n(\mathbf{x}) = \begin{cases} 1, & n = 0 \\ \exp\left(-\|\mathbf{x} - \mathbf{c}_n\|^2 / 2{\sigma_n}^2\right), & 1 \leq n \leq N \end{cases} \tag{3.2}$$

donde $\{\mathbf{c}_n\}$ son los centroides y $\{{\sigma_n}^2\}$ los correspondientes parámetros de dispersión. En la Sección 3.2 veremos cómo se pueden llevar a cabo la selección de estos parámetros.

El RAB estándar calculaba los pesos de salida $\{\alpha_t\}_{t=1}^T$ de modo que se minimizase

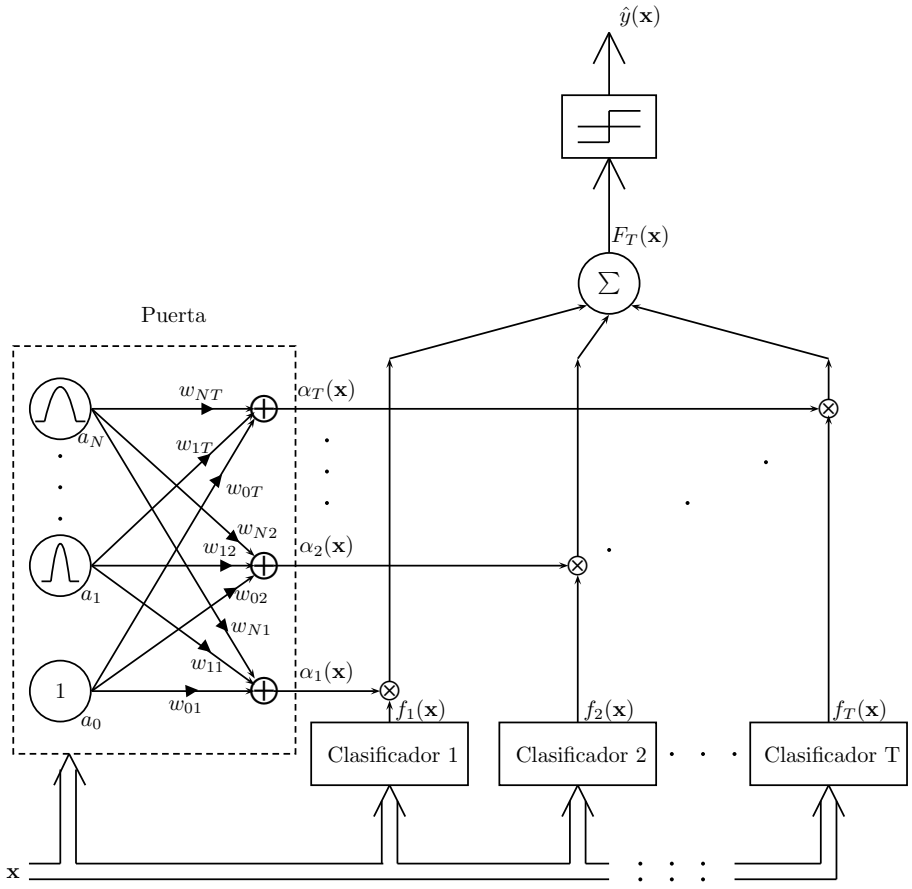


Figura 3.1: Arquitectura de un clasificador GCF-RAB.

la función de coste:

$$B_t = \frac{1}{L}\sum_{l=1}^{L}\exp[-F_t(\mathbf{x}^{(l)})y^{(l)}]. \tag{3.3}$$

Con el fin de retener las buenas propiedades de generalización del algoritmo RAB, en el algoritmo GCF-RAB entrenaremos los pesos de salida de la puerta, $\mathbf{w}_t$, siguiendo este mismo criterio, es decir,

$$\mathbf{w}_t \;=\; \arg\min_{\mathbf{w'}_t} B_t(\mathbf{w'}_t). \tag{3.4}$$

La salida parcial del sistema correspondiente a la t-ésima ronda viene dada por:

$$F_t(\mathbf{x}^{(l)}) = \sum_{t'=1}^{t} \alpha_{t'}(\mathbf{x}^{(l)}) f_{t'}(\mathbf{x}^{(l)}) = F_{t-1}(\mathbf{x}^{(l)}) + \mathbf{w'}_t^{\top} \mathbf{a}(\mathbf{x}^{(l)}) f_t(\mathbf{x}^{(l)}), \tag{3.5}$$

y (3.4) se puede reescribir como:

$$\mathbf{w}_t \;=\; \arg\min_{\mathbf{w'}_t} \frac{1}{L} \sum_{l=1}^{L} \exp\left[-y^{(l)} F_{t-1}(\mathbf{x}^{(l)})\right] \exp\left[-y^{(l)} \mathbf{w'}_t^{\top} \mathbf{a}(\mathbf{x}^{(l)}) f_t(\mathbf{x}^{(l)})\right] \tag{3.6}$$

De este modo, para minimizar (3.3) se aplica el algoritmo de descenso por gradiente estocástico

$$\mathbf{w}_t(l+1) = \mathbf{w}_t(l) - \eta \frac{\partial \exp\left[-y^{(l)} F_t(\mathbf{x}^{(l)})\right]}{\partial \mathbf{w}_t}$$

siendo el valor de la derivada igual a:

$$\frac{\partial \exp\left[-y^{(l)} F_t(\mathbf{x}^{(l)})\right]}{\partial \mathbf{w}_t} = -y^{(l)} f_t(\mathbf{x}^{(l)}) \exp\left[-y^{(l)} F_t(\mathbf{x}^{(l)})\right] \mathbf{a}(\mathbf{x}^{(l)}) \tag{3.7}$$

Por tanto, el vector de pesos, $\mathbf{w}_t$, se entrena para cada ronda t de acuerdo con la siguiente regla de actualización:

$$\mathbf{w}_t(l'+1) = \mathbf{w}_t(l') + \eta y^{(l')} f_t(\mathbf{x}^{(l')}) . \exp\left[-y^{(l')} F_t(\mathbf{x}^{(l')})\right] \mathbf{a}(\mathbf{x}^{(l')}) \tag{3.8}$$

donde η es el paso de adaptación del algoritmo y l' indica el indice de la muestra.

En la Tabla 3.1 se recoge el pseudocódigo que describe paso a paso el funcionamiento de este algoritmo.

3.2. Diseño de la puerta

La ventaja del algoritmo GCF-RAB frente a otros algoritmos Boosting radica en el esquema de fusión. Por tanto, el correcto diseño de la puerta es crucial para poder aprovechar todas las ventajas de esta nueva arquitectura.

El cuerpo de la puerta consta de un conjunto de funciones Gaussianas y su diseño involucra la selección de los centroides y el ajuste de su correspondiente parámetro de dispersión. La selección de estos parámetros va a realizarse de manera independiente, tal y como se describe a continuación.

Tabla 3.1: Pseudocódigo del algoritmo GCF-RAB.

1 - Entradas: $\{\mathbf{x}^{(l)}, y^{(l)}\}_{l=1}^{L}$

2 - Inicialización: $D_1(\mathbf{x}^{(l)}) = 1/L \quad \forall l$

4 - Selecciona: $\{\mathbf{c}_n, {\sigma_n}^2\}_{n=1}^{N}$

5 - Calcula: $\left\{a_n(\mathbf{x}^{(l)})\right\}_{n=0}^{N}$

3 - Para $t = 1, \ldots, T$

3.1 - Entrenar un clasificador, $f_t(\mathbf{x}^{(l)})$, minimizando la función de coste:

$$C_t = \sum_{l=1}^{L} D_t(\mathbf{x}^{(l)})[y^{(l)} - f_t(\mathbf{x}^{(l)})]^2.$$

3.2 - Entrenar el vector de pesos, $\mathbf{w}_t$, con un algoritmo de descenso por gradiente con regla de actualización:

$$\mathbf{w}_t(l'+1) = \mathbf{w}_t(l') + \eta y^{(l')} f_t(\mathbf{x}^{(l')}).\exp\left[-y^{(l')} F_t(\mathbf{x}^{(l')})\right] \mathbf{a}(\mathbf{x}^{(l')}).$$

3.3 - Calcular la salida de la puerta asociada a este clasificador como

$$\alpha_t(\mathbf{x}^{(l)}) = \mathbf{w}_t^{\top} \mathbf{a}(\mathbf{x}^{(l)}).$$

3.4 - Actualizar la función de énfasis para la siguiente iteración:

$$D_{t+1}(\mathbf{x}^{(l)}) = \frac{D_t(\mathbf{x}^{(l)}) \exp[-\alpha_t(\mathbf{x}^{(l)}) f_t(\mathbf{x}^{(l)}) y^{(l)}]}{Z_t},$$

$$\text{siendo } Z_t = \sum_{l=1}^{L} D_t(\mathbf{x}^{(l)}) \exp[-\alpha_t(\mathbf{x}^{(l)}) f_t(\mathbf{x}^{(l)}) y^{(l)}].$$

4 - El clasificador final implementa la función:

$$F_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t(\mathbf{x}) f_t(\mathbf{x})$$

siendo, por tanto, $\hat{y}(\mathbf{x}) = \text{sign}\,[F_T(\mathbf{x})]$ la clase estimada para el patrón $\mathbf{x}$.

3.2.1. Selección de centroides

Son muchos los procedimientos para seleccionar centroides que han sido propuestos por diferentes autores (en el Apartado 1.2.1 puede encontrase un resumen de al-

gunos de estos métodos). En esta Tesis Doctoral se propone seleccionar los centroides de las funciones de base radial siguiendo la idea propuesta en [Shin and Cho, 2002, Shin and Cho, 2003a, Shin and Cho, 2003b, Shin and Cho, 2007], donde se hace uso del algoritmo de los K vecinos más próximos (K–NN, K–"Nearest Neighbors") para evaluar la proximidad de las muestras a la frontera de clasificación y la facilidad con que éstas son clasificadas y, en función de estos parámetros, seleccionar los centroides.

Shin y Cho en [Shin and Cho, 2002] proponen aplicar el algoritmo K–NN sobre las muestras de entrenamiento con el fin de estimar la probabilidad de que la muestra $\mathbf{x}^{(l)}$ pertenezca a la clase j ($j \in \{\pm 1\}$),

$$P_j(\mathbf{x}^{(l)}) = \frac{K_j}{K},$$

siendo K es el número total de vecinos más próximos y K_j el número de miembros del vecindario que pertenecen a la clase j. Y a partir de esta probabilidad se definen las siguientes medidas:

1. La *proximidad* que presenta un dato, $\mathbf{x}^{(l)}$, a la frontera de decisión, que viene dada por la entropía de los valores P_j y se calcula como:

$$pr(\mathbf{x}^{(l)}) = \sum_j P_j(\mathbf{x}^{(l)}) \log_2 \frac{1}{P_j(\mathbf{x}^{(l)})}. \tag{3.9}$$

   Es obvio que este valor es mayor para las muestras cuyos vecinos tienen etiquetas variadas y, por lo tanto, será mayor para las muestras que se encuentran cerca de la frontera de clasificación, que tienen un valor de proximidad mayor que cero.

2. La *corrección*,

$$co(\mathbf{x}^{(l)}) = P_{y^{(l)}}(\mathbf{x}^{(l)}) \tag{3.10}$$

   que establece la probabilidad con la que una muestra $\mathbf{x}^{(l)}$ puede ser clasificada correctamente. Por lo tanto, un valor de corrección cercano a cero indica que la muestra tiene mayor probabilidad de estar ubicada del lado incorrecto de la frontera de clasificación.

A partir de estas dos medidas, podemos decir que las muestras que cumplan con las condiciones

$$pr(\mathbf{x}^{(l)}) > 0 \quad \text{y} \quad co(\mathbf{x}^{(l)}) \geq \frac{1}{2} \tag{3.11}$$

se encuentran cerca de la frontera y, además, son fáciles de clasificar. De este modo, tenemos un procedimiento razonable para preseleccionar un subconjunto de muestras como candidatas a ser utilizadas como centroides de las funciones Gaussianas.

Para completar la selección de los centroides y evitar incluir centroides redundantes entre sí por estar localizados en la misma región del espacio de observaciones, se aplica un procedimiento similar al algoritmo de clustering APC-III (Adaptive Pattern Classifier III) [Hwang and Bang, 1996]: para cada clase por separado, se elige como primer centroide la muestra preseleccionada que presenta mayor valor de proximidad según la ecuación (3.9). Una vez seleccionado el primer centroide, se eliminan del conjunto de muestras candidatas aquéllas que estén a una distancia menor que h del centroide que se acaba de seleccionar. Con el conjunto de muestras restantes, se repite este proceso hasta que no queden más muestras disponibles.

El valor de h se calcula como el promedio de la distancia mínima entre las muestras de entrenamiento multiplicado por un factor R,

$$h = R\frac{1}{L}\sum_{l=1}^{L}\left\|\mathbf{x}^{(l)} - \mathbf{x}_{\mathrm{NN}}^{(l)}\right\|_2, \tag{3.12}$$

donde $\mathbf{x}_{\mathrm{NN}}^{(l)}$ indica el vecino más cercano a $\mathbf{x}^{(l)}$.

El número total de centroides depende del número de muestras del subconjunto preseleccionado; a su vez, el número de muestras preseleccionadas depende del número de vecinos más próximos (K) al momento de calcular las medidas de *proximidad* y *corrección*. En consecuencia, es necesario ajustar el valor de K de un modo razonable.

3.2.2. Selección de los parámetros de dispersión

El criterio empleado para la selección del parámetro de dispersión de las funciones de base radial puede dar lugar a diferentes variantes del algoritmo CGF-RAB. En esta Tesis Doctoral se exploran dos posibles estrategias:

- La primera de ellas selecciona como parámetros de dispersión valores proporcionales al promedio de la distancia euclídea entre $\mathbf{c}_n$ y todas las muestras que están más cerca de él que del resto de centroides, es decir,

$$\sigma_n = r\frac{1}{\#C_n}\sum_{\mathbf{x}^{(l)}\in C_n}\left\|\mathbf{x}^{(l)}-\mathbf{c}_n\right\|_2 \tag{3.13}$$

  donde C_n es el conjunto de muestras pertenecientes al centroide n-ésimo , $\#$ indica su cardinalidad y r es el factor de escala.

- La segunda estrategia utiliza directamente como parámetro de dispersión el valor del radio de la esfera (h) del algoritmo APC-III dado en la ecuación (3.12).

3.3. Versiones del algoritmo GCF-RAB

Tal como se explicó en el Apartado 3.2, el diseño de la puerta involucra la selección de los centroides y el ajuste de sus correspondientes parámetros de dispersión. Los procedimientos empleados para tales tareas suponen cuatro versiones del algoritmo GCF-RAB.

1. La primera versión, a la que denominaremos S1, fija el parámetro de dispersión según la primera estrategia expuesta en el Apartado 3.2.2. Por lo tanto, los parámetros a ajustar son el número de vecinos más próximos K, el factor de escala R para ajustar el valor de h, y el factor de escala r para la selección de σ_n.

2. La segunda versión, que denominamos S2, selecciona el valor del parámetro de dispersión según la segunda estrategia explicada en el Apartado 3.2.2. En consecuencia, los parámetros a ajustar son K y R.

3. Las dos restantes, son versiones reducidas de S1 y S2, que denominaremos S1(R) y S2(R). En estas dos versiones, el número de vecinos más próximos K se elije según el procedimiento empírico propuesto en [Shin and Cho, 2003b]. Este procedimiento consiste en aplicar, en primer lugar, un clasificador 1-NN sobre las muestras de entrenamiento. Una vez obtenida la solución 1-NN, el valor de K se incrementa iterativamente hasta que el número de muestras preseleccionadas siguiendo las condiciones de (B.8) sea mayor o igual que $2L\hat{P}_e$, donde $\hat{P}_e$ es la probabilidad de error calculada con un clasificador 1-NN (los detalles de este procedimiento se encuentran recogidos en el Apéndice B). Este procedimiento para fijar K permite reducir el esfuerzo computacional en la fase de diseño.

3.4. Pruebas experimentales

Las prestaciones del algoritmo propuesto, GCF-RAB, en sus diferentes versiones, se analizan en comparación con los resultados proporcionados por el algoritmo RAB estándar. Se analizarán su capacidad de generalización, el coste computacional de clasificar un nuevo dato, y la sensibilidad a los diferentes parámetros propios de la arquitectura propuesta. Antes de ello, se detallan las bases de datos y la configuración de los experimentos.

3.4.1. Bases de datos

Para la evaluación de los métodos propuestos se han considerado ocho base de datos con diferentes características (tamaño, dimensión y dificultad). Dos de ellas

son problemas sintéticos, Kwok [Kwok, 1999] y Ripley [Ripley, 1994]. Las restantes bases de datos son problemas reales; cinco de ellas se han obtenido del repositorio de la UCI [Frank and Asuncion, 2010] (Abalone, Breast, Contraceptive, Hepatitis, y Ionosfera), y la última, Crabs, perteneciente a [Ripley, 1996].

En la Tabla 3.2 se resumen las principales características de cada uno de estos problemas (d: dimensión; L_1/L_{-1}: número de muestras) para los conjuntos de entrenamiento y test. La notación o abreviatura con la que se va a denotar cada problema es con sus dos primeras letras. En el Apéndice C se presenta una descripción más detallada de cada uno de estos problemas.

Tabla 3.2: Principales características de las bases de datos.

Problemas	**Dimensión** d	**Nº. Muestras entrenamiento** L_1/L_{-1}	**Nº. Muestras test** L_1/L_{-1}
Abalone (Ab)	8	1238/1269	843/827
Breast (Br)	9	145/275	96/183
Contraceptive (Co)	9	506/377	338/252
Crabs (Cr)	7	59/61	41/39
Hepatitis (He)	19	70/23	53/9
Ionosfera (Io)	34	101/100	124/26
Kwok (Kw)	2	300/200	6120/4080
Ripley (Ri)	2	125/125	500/500

3.4.2. Entrenamiento de los conjuntos a evaluar

Para el diseño de los conjuntos RA y GCF-RAB, se emplean como clasificadores base redes neuronales de tipo MLP. En el siguiente subapartado se indican los parámetros de diseño y de entrenamiento empleados para llevar a cabo estos experimentos.

El entrenamiento de los conjuntos se realiza según indica el pseudocódigo recogido en las Tablas 2.1 ó 3.1, según se trate del algoritmo RA o GCF-RAB, respectivamente,

en los que se supone conocido el número de clasificadores base que constituyen el conjunto, T. Más adelante, se describe cómo se lleva a cabo la selección de T durante el entrenamiento de los conjuntos.

Diseño de los clasificadores base

La arquitectura de los MLPs empleados consiste en una única capa oculta, cuyo número de neuronas denotamos por M. En cuanto a la función de coste para el entrenamiento, en ambos casos se entrenan para que minimice

$$C_t = \sum_{l=1}^{L} D_t(l) \left[y^{(l)} - f_t(\mathbf{x}^{(l)}) \right]^2, \tag{3.14}$$

donde $D_t(\cdot)$ es la función de énfasis convencional de RAB.

Para ello, se inicializan aleatoriamente las componentes de los pesos del MLP con valores uniformemente distribuidos en el intervalo [-0.2, 0.2] , y se realiza su actualización mediante el algoritmo de retropropagación. El paso de aprendizaje se inicializa en 0.01 (tanto para la capa de entrada como para la de salida) y decrece linealmente hasta 0 durante las 100 épocas que dura el entrenamiento (habiéndose comprobado que este número de épocas es suficiente para asegurar la convergencia).

Junto con el algoritmo de retropropagación, se emplea un algoritmo de detención prematura del entrenamiento, utilizando el 80 % de los datos disponibles para entrenar el MLP y el 20 % restante para validar el diseño, es decir, para elegir los pesos correspondientes a la época que presenta un menor error sobre esta partición, reduciendo así los efectos perniciosos del sobreajuste.

Diseño de la puerta

Con respecto a los pesos de la puerta, tal como se explicó en el Apartado 3.1, se ajustan con una busqueda por gradiente. Para ello, se inicializan de forma aleatoria con valores en el intervalo [-0.05, 0.05] uniformemente distribuidos, y el paso de

aprendizaje η para la actualización:

$$\mathbf{w}_t(l'+1) = \mathbf{w}_t(l') + \eta y^{(l')} f_t(\mathbf{x}^{(l')}).\exp\left[-y^{(l')} F_t(\mathbf{x}^{(l')})\right] \mathbf{a}(\mathbf{x}^{(l')}) \tag{3.15}$$

se fija en 0.01 durante las 100 épocas necesarias para garantizar la convergencia.

Selección del número de clasificadores base

Otro de los aspectos a considerar en el diseño es el número total de clasificadores base a emplear, es decir, el número total de rondas, T. Aunque se podría pensar en aplicar el procedimiento de Validación Cruzada (CV) para seleccionar este parámetro, o atender a la evolución del error de entrenamiento, ninguna de estas soluciones permite obtener una buena generalización. Aplicar un proceso de CV puede ocasionar que la velocidad de convergencia se modifique al reducir el número de muestras de entrenamiento, llegando a resultados de convergencia distintos de los óptimos. Si, por el contrario, se atendiese a la evolución del error de entrenamiento, no habría garantía de que el error de generalización presentase un comportamiento similar.

Por las razones anteriores, para el caso del algoritmo RAB se detiene el crecimiento del conjunto atendiendo la evolución de los valores de α_t de acuerdo con el criterio propuesto en [Gómez-Verdejo et al., 2008]. Dicho criterio consiste en detener el crecimiento del conjunto cuando se cumple que

$$\frac{\sum_{t=T-9}^{T} \alpha_t}{\sum_{t=1}^{T} \alpha_t} < T_{stop}, \tag{3.16}$$

donde T_{stop} se fija de manera empírica a 0.1.

En el caso del algoritmo GCF-RAB, la evolución de α_t depende de los datos de entrada; por tanto, no es una buena medida para considerarla en el criterio de parada. Sin embargo, otro parámetro que provee información sobre su crecimiento es el parámetro de separación del conjunto en la t-ésima ronda

$$\gamma'_t = \sum_{l=1}^{L} D_t(l) F_t(\mathbf{x}^{(l)}) d^{(l)}, \tag{3.17}$$

que, cuando los clasificadores base no son capaces de clasificar las muestras más enfatizadas (erróneas), toma valores pequeños que apenas cambian. Teniendo en cuenta lo anterior, para el algoritmo GCF-RAB, el crecimiento del conjunto se detiene cuando se cumple que

$$\frac{\sum_{t=T-9}^{T} \gamma'_t}{\sum_{t=1}^{T} \gamma'_t} < T'_{stop} \tag{3.18}$$

con T'_{stop} fijado empíricamente a 0.3.

Selección de los parámetros de diseño

Los parámetros de diseño se han seleccionado mediante un proceso de validación cruzada de 5 particiones en el que se han empelado 10 repeticiones sobre cada partición.

Ambos algoritmos, RAB y GCF-RAB, requieren la selección del número de neuronas en la capa oculta de los MLPs, M.

La versión S1 necesita seleccionar tres parámetros adicionales: el número de vecinos más próximos, K, junto con los factores de escala R y r. La versión S2 requiere dos parámetros adicionales, K y R. Las versiones reducidas, S1(R) y S2(R), no requieren del ajuste del parámetro K. En consecuencia, los parámetros adicionales a ajustar para S1(R) son los factores de escala R y r; sólo R para la versión S2(R).

En todos los casos, el rango de exploración para cada uno de los parámetros se ha fijado de tal manera que los valores óptimos no estén ubicados en los extremos en el momento de evaluar la evolución del error de validación.

Según lo dicho, el valor de M se explora en el intervalo [2, 10] en pasos unitarios, siendo necesario extender este rango hasta 60 sólo para el caso del problema Ri. El parámetro K se explora en seis valores pares: 2, 4, 6, 8, 10 y 12. Respecto a los factores de escala, R se selecciona entre 20 valores equidistantes en el intervalo [1, 50], y r entre 10 valores linealmente espaciados en el intervalo [0.5, 5].

Según lo anterior, el método propuesto en su versión completa (S1) requiere del ajuste de cuatro parámetros, lo que podría ocasionar inestabilidad en la validación.

Sin embargo, el método es bastante robusto a la selección de sus parámetros, tal como se comprobará en el Apartado 3.4.5.

3.4.3. Resultados

En la Tabla 3.3 se muestran las tasas de error de clasificación en test, CE, obtenidas después de promediar sobre 50 repeticiones para cada conjunto usando los valores de los parámetros seleccionados por CV. También se indica, por un lado, el tamaño medio (T) de los conjuntos, y, por otro lado, el p-valor (p) proporcionado por el t-test aplicado a los errores de clasificación de RAB y GCF-RAB (mayor descripción sobre el t-test véase en el Apéndice D). Además, para facilitar la comparación entre los algoritmos y las diferentes variantes, se resalta en negrita el mejor resultado (el que presenta el menor error de clasificación) y los valores de p por debajo de 0.05, lo que indica diferencia estadística entre los resultados con un nivel de certeza del 95 %.

Analizando los resultados de la Tabla 3.3 pueden extraerse las siguientes conclusiones:

- En todos los casos, el algoritmo GCF-RAB en cualquiera de sus variantes ofrece mejores resultados que el RAB convencional, salvo en Cr y Kw; en que ambos algoritmos empatan.

- Como se esperaba, la variante S1 (versión completa de GCF-RAB) ofrece los mejores resultados, superando con una diferencia significativa (p <0.05) al RAB en la mayoría de los casos, salvo empates para Cr y Kw. De hecho, esta ventaja es muy clara para el problema Ri, significativa para Co, y ligera para los problemas Ab, Br, He, y Io.

- Si analizamos los casos Cr y Kw, donde los algoritmos GCF-RAB y RAB, presentan resultados equivalentes, podemos observar:

Tabla 3.3: Tasas de error de clasificación (CE) y tamaño medio de los conjuntos (T) para los algoritmos RAB y GCF-RAB obtenidos después de promediar sobre 50 repeticiones. También se muestra el p-valor proporcionado por el t-test.

		RAB	**GCF-RAB**			
			S1	**S2**	**S1(R)**	**S2(R)**
Ab	$CE(\%)$	19.4 ± 0.02	**19.1** ± 0.3	19.3 ± 0.3	**19.1** ± 0.3	19.3 ± 0.3
	T	31.2 ± 0.4	16.3 ± 0.4	30.5 ± 0.9	19.0 ± 0.4	29.2 ± 0.8
	p	—	**0**	**0.04**	**0**	0.11
Br	$CE(\%)$	2.6 ± 0.4	**2.2** ± 0.3	2.3 ± 0.4	2.4 ± 0.3	2.4 ± 0.5
	T	21.3 ± 4.2	16.1 ± 0.8	18.0 ± 1.2	18.2 ± 0.7	32.6 ± 1.0
	p	—	**0**	**0**	**0.02**	0.11
Co	$CE(\%)$	29.0 ± 0.2	**27.4** ± 0.8	28.1 ± 0.8	28.2 ± 0.8	28.2 ± 0.8
	T	33.7 ± 0.7	21.7 ± 1.1	29.7 ± 0.7	29.9 ± 1.4	30.4 ± 1.2
	p	—	**0**	**0**	**0**	**0**
Cr	$CE(\%)$	**2.5** ± 0.0	**2.5** ± 0.0	**2.5** ± 0.0	**2.5** ± 0.0	**2.5** ± 0.0
	T	11.1 ± 0.8	16.1 ± 0.5	8.1 ± 0.2	19.3 ± 0.6	31.5 ± 0.9
	p	—	1	1	1	1
He	$CE(\%)$	8.9 ± 1.8	**8.1** ± 1.9	8.4 ± 1.8	8.5 ± 1.5	8.5 ± 1.7
	T	22.2 ± 3.9	15.6 ± 1.8	18.7 ± 2.5	16.9 ± 1.9	27.5 ± 2.0
	p	—	**0.02**	0.13	0.14	0.31
Io	$CE(\%)$	4.5 ± 0.9	**4.0** ± 1.0	4.2 ± 1.7	4.1 ± 1.5	**4.0** ± 1.6
	T	22.2 ± 2.4	17.3 ± 1.7	24.9 ± 1.9	19.1 ± 5.4	26.9 ± 3.6
	p	—	**0.01**	0.14	0.09	**0.05**
Kw	$CE(\%)$	**11.7** ± 0.01	**11.7** ± 0.2	**11.7** ± 0.1	**11.7** ± 0.2	**11.7** ± 0.2
	T	29.3 ± 0.1	19.8 ± 1.1	29.0 ± 1.2	28.8 ± 1.2	36.3 ± 1.4
	p	—	0.7	0.94	0.86	0.53
Ri	$CE(\%)$	9.7 ± 0.01	**8.5** ± 0.2	8.6 ± 0.3	9.1 ± 0.2	9.2 ± 0.3
	T	28.9 ± 0.2	17.9 ± 0.6	25.6 ± 0.5	24.2 ± 2.3	33.4 ± 0.7
	p	—	**0**	**0**	**0**	**0**

- El valor de la varianza igual a cero para Cr parece indicar que los errores corresponden a muestras atípicas (outliers) que no pueden ser clasificadas correctamente por ninguno de los conjuntos.

- Kw es un problema artificial que muestra una frontera de clasificación de Bayes relativamente suave, y el carácter local de GCF-RAB no otorga ventaja en este tipo de problemas. Por otra parte, los resultados son cercanos al error teórico de un clasificador Bayesiano, que esta en 11.3 %.

- Respecto a las variantes del algoritmo GCF-RAB, la versión S2 ofrece resultados ligeramente peores que la versión S1, pero en ningún caso peores que RAB. Esta reducción en la precisión del algoritmo se puede atribuir al hecho que S2 restringe los valores del los parámetros de dispersión de las funciones Gaussianas al mismo valor de h. Las versiones reducidas S1(R) y S2(R) proporcionan, en general, peores prestaciones que las versiones S1 y S2, respectivamente (sólo en el caso de Io S2(R) proporciona mejores resultados que S2). Estas desventajas han de atribuirse a las restricciones impuestas en la selección del parámetro K.

- Cabe resaltar que, analizando los resultados del t-test y a la luz de la Tabla 3.3, los valores de p apoyan plenamente las diferencias significativas a favor del algoritmo GCF-RAB en su versión completa S1.

Aparte de lo anterior, cabe mencionar que el algoritmo GCF-RAB obtiene mayores ventajas en problemas que presentan cambios bruscos en su frontera de clasificación, tales como Ri y Co. Para apoyar esta afirmación, en la Figura 3.2 se muestran las fronteras de clasificación obtenidas con los diseños finales de los algoritmos RAB y GCF-RAB para un problema típico, Ri. También se muestra la frontera de clasificación de Bayes. Nótese que las capacidades locales de GCF-RAB (S1) permiten obtener una mayor cercanía a la frontera de Bayes en las regiones donde se concentran una mayor cantidad de muestras de entrenamiento, mientras que RAB tiene dificultades en acercarse a la frontera de Bayes donde ésta presenta cambios bruscos.

En definitiva, se puede concluir que, en general, la idea de introducir una puerta para combinar las salidas de los clasificadores base bajo una formulación RAB sirve para mejorar las prestaciones de los diseños resultantes. Obviamente, esto no es gratis, ya que el entrenamiento de los pesos de la puerta y, especialmente, el costoso proceso de CV aumentan en gran medida el esfuerzo computacional de diseño (en varios órdenes de magnitud). Este es el precio a pagar para mejorar de los excelentes

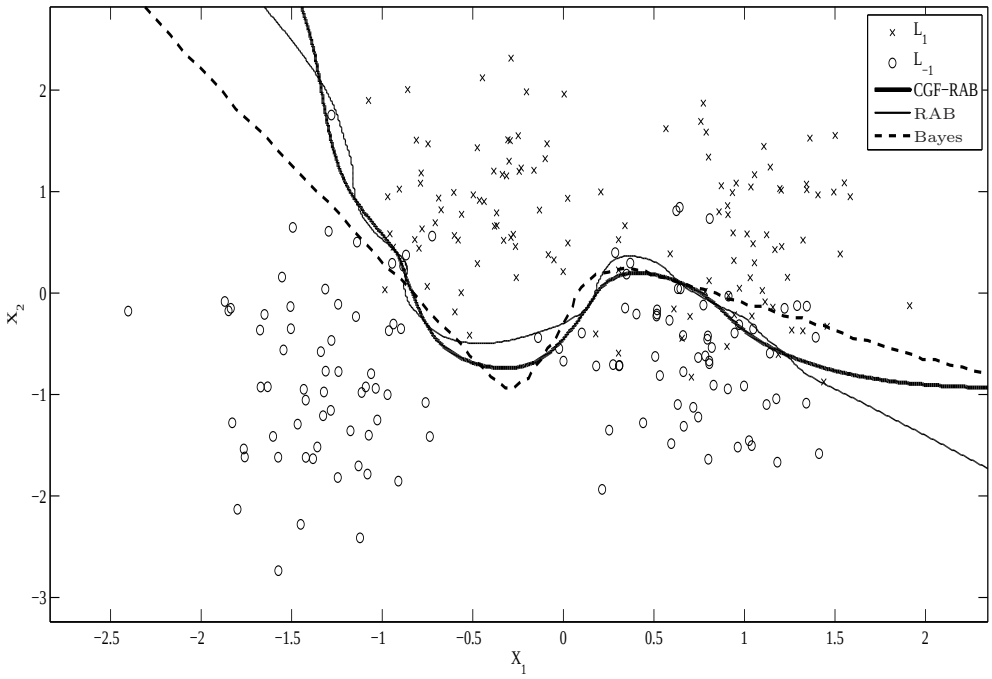


Figura 3.2: Frontera de clasificación típica proporcionada por los algoritmos RAB y GCF-RAB (S1) en el problema Ri.

resultados que ya de por si otorga el algoritmo RAB. Sin embargo, en la Tabla 3.3 se puede observar que, en muchos casos, los diseños GCF-RAB requieren un menor número (promedio) de clasificadores base y un menor número de neuronas ocultas para los MLPs.

3.4.4. Carga computacional en la fase de operación

Dado que cada elemento del conjunto GCF-RAB no sólo contiene el MLP correspondiente, sino que además incluye la puerta, es necesario realizar cálculos más detallados para comprobar si además de mejores prestaciones, también se obtiene ventaja computacional en la fase de operación, es decir, cuando se utilizan los conjuntos para clasificar un nuevo dato.

Tabla 3.4: Valores de los parámetros no entrenables obtenidos por el proceso de CV para los algoritmos RAB y GCF-RAB.

	RAB	**GCF-RAB**			
		S1	**S2**	**S1(R)**	**S2(R)**
	M	$M/N/K/R/r$	$M/N/K/R$	$M/N/K/R/r$	$M/N/K/R$
Ab	4	2/50/6/6.2/1	2/7/6/34.5	2/7/4/34.5/1	2/7/4/34.5
Br	6	2/18/8/11.3/3.5	2/18/2/1.0	2/4/3/24.2/1	6/7/3/11.3
Co	2	2/165/3.6/3.5	2/8/4/34.5	2/48/2/11.3/1	2/17/2/24.2
Cr	2	2/34/8/1.0/1.5	2/11/2/1.0	2/7/2/1.0/1	2/7/2/1.0
He	17	4/20/4/3.6/4	4/3/2/21.6	6/2/2/39.7/0.5	4/2/2/31.9
Io	5	2/10/6/34.5/3	2/66/10/1.0	2/8/2/26.8/2.5	6/21/2/6.2
Kw	15	2/4/6/31.9/2	2/7/10/16.5	4/4/2/31.9/1	4/4/2/42.3
Ri	48	2/31/10/1.0/2	6/10/4/6.2	6/5/2/21.6/0.5	2/6/2/16.5

Una estimación razonable del esfuerzo computacional en la fase de operación de los algoritmos RAB y GCF-RAB puede realizarse mediante el cálculo del correspondiente número de multiplicaciones y, si no se hace uso de una tabla con los valores de las transformaciones no lineales precalculadas para obtener la salida de las transformaciones no lineales, el número de tales transformaciones.

Con respecto al número de multiplicaciones, debemos tener en cuenta que:

- para obtener el argumento de cada elemento del cuerpo de la puerta $\mathbf{a}(\mathbf{x})$, son necesarias d multiplicaciones para el cálculo de la distancia al cuadrado, y el resultado tiene que ser multiplicado por $1/2\sigma_n^2$. Por lo tanto, para este cálculo se necesitan $N(d+1)$ multiplicaciones, siendo N el número de elementos de la puerta. Para obtener la salida de cada puerta, son necesarias N multiplicaciones adicionales.

- Cada MLP con M neuronas ocultas requiere de Md (capa de entrada) más M (capa de salida) multiplicaciones.

- Además de lo anterior, para la arquitectura RAB estándar en cada ronda es necesaria una multiplicación adicional (para α_t). Del mismo modo para GCF-RAB, si añadimos las salidas de los elementos de la puerta antes de multiplicar.

En consecuencia, si T es el número de clasificadores base, la arquitectura RAB necesita $T(M(d+1)+1)$ multiplicaciones, y la arquitectura GCF-RAB requiere de $T(M(d+1)+N+1)+N(d+1)$ multiplicaciones.

Con respecto al número de transformaciones no lineales, si no se distingue entre transformaciones exponenciales y sigmoidales (lo cual va a favor del RAB estándar), el número de elementos que se incluyen en cada máquina conjunta es de $T(M+1)$ para RAB y $T(M+1)+N$ para GCF-RAB.

En la Tabla 3.5 se muestran los valores promedios correspondientes al número de multiplicaciones y al número de funciones no lineales empleados por los algoritmos RAB y GCF-RAB para clasificar un nuevo dato. Cabe resaltar que, si se emplea una tabla con los valores de las transformaciones no lineales precalculadas, los resultados mostrados a la izquierda de la Tabla 3.5 indican la carga computacional.

Tabla 3.5: Número de multiplicaciones y número de funciones no lineales para clasificar un nuevo dato por los conjuntos RAB y GCF-RAB en sus diferentes versiones.

	A. Número de multiplicaciones					**B. Número de funciones no lineales**				
	RAB	**GCF-RAB**				**RAB**	**GCF-RAB**			
		S1	**S2**	**S1(R)**	**S2(R)**		**S1**	**S2**	**S1(R)**	**S2(R)**
Ab	1154.4	1574.7	856.0	**557.0**	882.2	156.0	98.9	98.5	**64.0**	94.0
Br	1299.3	807.9	882.0	**495.0**	2286.8	149.1	66.3	72.0	**58.6**	235.2
Co	**707.7**	5686.2	941.3	2543.1	1325.2	101.1	230.1	**97.1**	137.7	108.2
Cr	**188.7**	1093.1	314.8	519.2	812	**33.3**	82.3	35.3	64.9	101.5
He	7570.2	1975.6	**1630.8**	2118.7	2322.5	399.6	98.0	**96.5**	120.3	139.5
Io	3907.2	**1751.3**	5721.3	1788.9	6975.8	132.2	**61.9**	140.7	65.3	209.3
Kw	1347.8	**229.8**	427.0	501.6	629.1	468.8	**63.4**	94.0	148.0	185.5
Ri	4190.5	773.2	772.4	595.8	**452.1**	1416.1	**84.7**	189.2	174.4	106.2

Analizando los números de multiplicaciones mostrados en la Tabla 3.5, se puede decir que en la mayoría de casos el método propuesto utiliza un menor número de multiplicaciones que RAB, con dos únicas excepciones para los casos Co y Cr,

donde RAB necesita un menor número de multiplicaciones. Respecto al número de funciones no lineales, en la mayoría de los casos GCF-RAB realiza un menor número de transformaciones no lineales que RAB, salvo el caso de Cr, donde RAB requiere menos transformaciones no lineales.

De acuerdo con lo anterior, se podría resaltar que, a pesar de incluir una red adicional como es la puerta, existe una relativa ventaja computacional en operación del esquema GCF-RAB sobre el RAB estándar. Por lo tanto, el algoritmo GCF-RAB ofrece la posibilidad de obtener máquinas con un mejor rendimiento y con un esfuerzo computacional de clasificación menor que el algoritmo RAB estándar.

3.4.5. Análisis de los problemas de sensibilidad

Aunque los diseños propuestos presentan prestaciones excelentes, es necesario evaluar las limitaciones que introduce el proceso de CV, especialmente en este caso, en donde el diseño de la máquina involucra varios parámetros libres a explorar. De este modo, se podrá analizar la robustez del método con respecto a los parámetros correspondientes.

Aunque estudiar la sensibilidad de los diseños propuestos con respecto a todos sus parámetros (M, K, h o N para S1 y S2, y adicionalmente r para S1 y S1(R)) no es una tarea fácil, se puede utilizar el concepto de "máquina omnisciente" para llevar a cabo una versión simplificada, pero esclarecedora, de este análisis.

Obviamente, una "máquina omnisciente" no es un diseño válido: es una máquina en la que los parámetros no entrenables se seleccionan de acuerdo con los resultados sobre las muestras del conjunto de test. Pero si existen pequeñas diferencias entre los valores de los parámetros correspondientes y los valores obtenidos a partir de CV, se podría decir que el proceso de CV ha sido completamente exitoso. Es más, si existen diferencias, pero las prestaciones son similares, es un síntoma de dificultades en el proceso de CV debido a que la superficie del error, considerada como una función de los parámetros, es relativamente plana. En este caso, los diseños resultantes se

Tabla 3.6: Prestaciones presentadas por el algoritmo GCF-RAB en su versión S1 con la aproximación "omnisciente".

		GCF-RAB	
		S1	**Omnisciente**
		$M/N/K/R/r$	$M/N/K/R/r$
Ab	$CE(\%)$	19.1 ± 0.3	19.0 ± 0.4
	T	16.3 ± 0.4	16.0 ± 0.5
	param.	2/50/6/6.2/1	2/405/10/1.0/2.5
Br	$CE(\%)$	2.2 ± 0.3	2.2 ± 0.3
	T	16.1 ± 0.8	16.4 ± 1.3
	param.	2/18/8/11.3/3.5	2/10/4/11.3/3.5
Co	$CE(\%)$	27.4 ± 0.8	27.4 ± 0.5
	T	21.7 ± 1.1	21.4 ± 0.5
	param.	2/165/3.6/3.5	2/21/4/21.6/3.0
Cr	$CE(\%)$	2.5 ± 0.0	2.5 ± 0.0
	T	16.1 ± 0.5	16.1 ± 0.4
	param.	2/34/8/1.0/1.5	2/11/2/1.0/1
He	$CE(\%)$	8.1 ± 1.9	7.7 ± 1.3
	T	15.6 ± 1.8	16.1 ± 1.1
	param.	4/20/4/3.6/4	4/2/6/26.8/2
Io	$CE(\%)$	4.0 ± 1.0	3.8 ± 1.8
	T	17.3 ± 1.7	17.7 ± 4.3
	param.	2/10/6/34.5/3	4/66/10/1.0/2.5
Kw	$CE(\%)$	11.7 ± 0.2	11.7 ± 0.1
	T	19.8 ± 1.1	20.5 ± 1.1
	param.	2/4/6/31.9/2	2/10/4/11.3/3.5
Ri	$CE(\%)$	8.5 ± 0.2	8.4 ± 0.3
	T	17.9 ± 0.6	17.8 ± 0.5
	param.	2/31/10/1.0/2	6/31/10/1.0/2

pueden considerar aceptables y, además se puede decir que el método es robusto frente a dichos parámetros, ya que pequeños cambios en los valores de los parámetros apenas modifican las prestaciones del algoritmo.

En la Tabla 3.6 se muestran las tasas del error de clasificación y los valores de los parámetros correspondientes a los diseños omniscientes para el algoritmo GCF-RAB en la versión S1, que es la que más parámetros libres tiene y, por tanto, la más sensible. A la vista de los resultados, está claro que existen diferencias entre los valores de los parámetros obtenidos por CV y la aproximación omnisciente; por lo

tanto, no se puede hablar de un proceso de CV absolutamente exitoso. Sin embargo, también es evidente que las diferencias en prestaciones para los problemas Br, Co, Cr, y Kw, son irrelevantes, casi inapreciables para Ab y Ri, y poco relevantes para los casos He y Io. Luego podemos concluir que los métodos de diseño que se han seguido para la arquitectura GCF-RAB muestran un grado de robustez satisfactorio.

3.4.6. Convergencia

Otro aspecto importante a considerar del comportamiento del algoritmo GCF-RAB es la convergencia: dado que se incluye un esquema de fusión entrenable, conviene comprobar que la convergencia tiene lugar sin problemas.

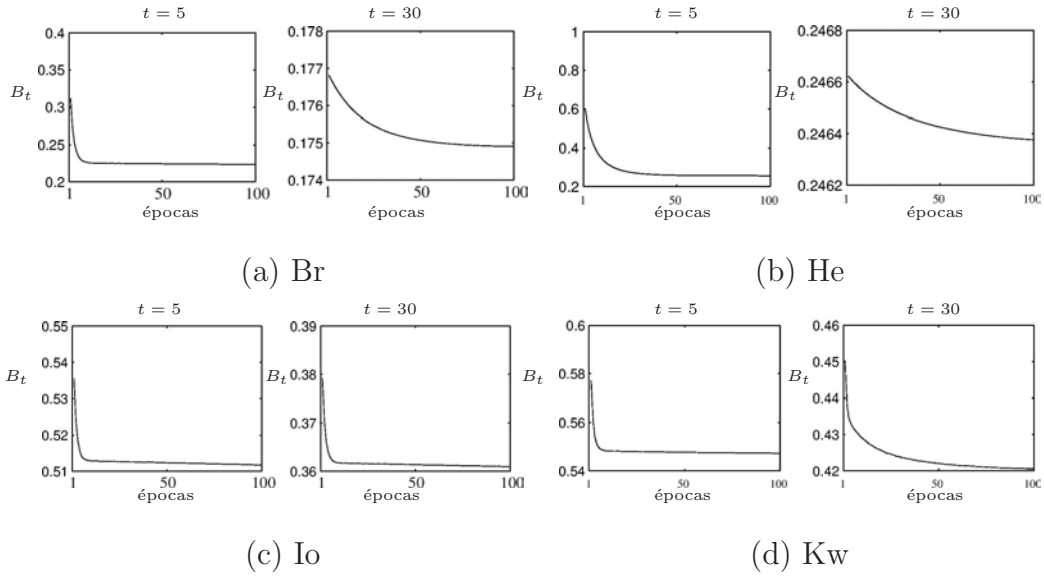
Para analizarlo, en primer lugar, en la Figura 3.3 se muestra la evolución de B_t con el número de épocas durante el ajuste del vector de pesos $\mathbf{w}_t$ correspondiente a la t-ésima ronda. Se consideran cuatro casos representativos (Br, He, Io y Kw) y la versión S1. Las curvas mostradas se han promediado sobre 50 repeticiones y corresponden a los instantes $t = 5$ y $t = 30$, con el fin de visualizar el comportamiento en un rango apreciable.

Como era de esperar, B_t decae en diferentes formas, pero siempre hasta que los valores de B_t cambian muy poco y se alcanza la condición de parada. Por tanto, podemos concluir que no se observan problemas de convergencia: en todos los casos se consigue minimizar B_t a medida que se van agregando aprendices.

Por otro lado, en la Figura 3.4 se muestra la evolución del error de clasificación, CE, obtenido a la salida del conjunto en función del número de rondas (T) y promediado sobre 50 repeticiones, para los algoritmos RAB y GCF-RAB en sus versiones S1 y S2. También se incluyen los puntos de parada dados por la condición $\frac{\sum_{t=T-9}^{T}\gamma'_t}{\sum_{t=1}^{T}\gamma'_t} < 0.3$.

A la vista de estas curvas, podemos decir que, de manera general, no se aprecian problemas de convergencia y las condiciones de parada resultan razonables.

En la mayoría de los casos, el algoritmo GCF-RAB presenta una velocidad de



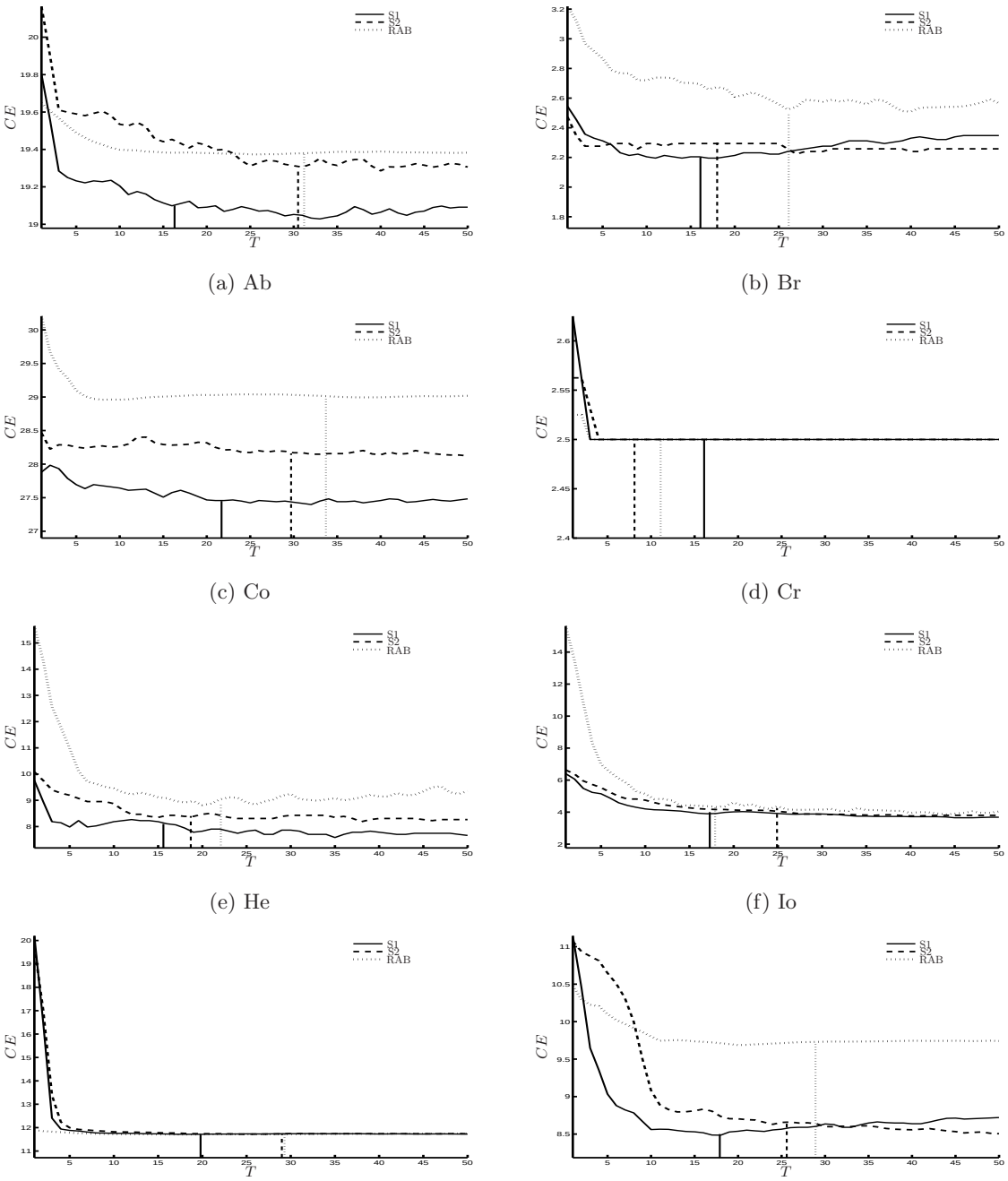
(a) Br (b) He

(c) Io (d) Kw

Figura 3.3: Evolución de la función de coste exponencial del margen (B_t) con el número de épocas para el algoritmo GCF-RAB (versión S1) en los problemas Br, He, Io y Kw, promediadas sobre 50 repeticiones.

convergencia más rápida que RAB, con excepción de los problemas Co y Kw, donde GCF-RAB comienza con un CE mayor que el de RAB. Sin embargo, en pocas rondas consigue descender hasta alcanzar un comportamiento sin cambios cuando se siguen agregando aprendices.

Tampoco se observa la aparición de problemas de sobreajuste salvo en los registros Br y Ri, que cuando se entrenan con la versión S1 presentan una tendencia al sobreajuste. Esto se debe, probablemente, a la potencia expresiva de las combinaciones con puerta de los clasificadores base. En consecuencia, es necesario vigilar el sobreajuste para estos casos durante el proceso de diseño del conjunto GCF-RAB.



(a) Ab (b) Br

(c) Co (d) Cr

(e) He (f) Io

(g) Kw (h) Ri

Figura 3.4: Curvas de evolución del error de clasificación, CE, promediadas sobre 50 repeticiones, en función del número de rondas, T, para los algoritmos RAB y GCF-RAB en sus versiones S1 y S2, para las ocho bases de datos utilizadas. También se muestran los puntos de parada.

3.5. Conclusiones

Para evitar la limitación debida a que Real AdaBoost (RAB) restringe la fusión de sus aprendices a una combinación lineal, en este capítulo, tomando como base el esquema de fusión que caracteriza las Mezclas de Expertos, se ha descrito un nuevo esquema de fusión que involucra puertas locales con estructuras de núcleos para mejorar las prestaciones de RAB sin alterar el entrenamiento de los aprendices.

Las ventajas de esta propuesta, denominada RAB con Fusión Controlada por Puerta (GCF-RAB), se han comprobado experimentalmente para una serie de procedimientos de diseño diferentes, obteniendo sistemáticamente resultados positivos en prestaciones y en carga computacional para clasificar un nuevo dato . De esta manera, se presenta un método más para hacer frente a las limitaciones intrínsecas de RAB.

El precio por mejorar las buenas prestaciones del algoritmo RAB, es un aumento importante en el esfuerzo computacional durante la fase de diseño, principalmente debido a las exploraciones de validación cruzada. En todo caso, los procedimientos empleados muestran la debida robustez.

Capítulo 4

Boosting con aprendices tipo SVM

Otra de las características de los algoritmos Boosting es el uso de aprendices débiles. Por ello, generalmente el conjunto es una arquitecturas de gran tamaño, debido a la poca capacidad expresiva de los aprendices. Trabajos posteriores al original han utilizado clasificadores considerados como fuertes, por ejemplo MLPs [Gómez-Verdejo et al., 2006, Gómez-Verdejo et al., 2008, Schwenk and Bengio, 2000, Mayhua-López et al., 2010, Mayhua-López et al., 2012] o RBFs [Rätsch et al., 2001], consiguiendo estructuras más compactas (con menor número de aprendices) y, además, con mejores prestaciones.

Siguiendo esta línea, parece que el uso de SVMs [Schölkopf and Smola, 2002, Shawe-Taylor and Cristianini, 2004, Vapnik, 1995] como clasificadores base podría proporcionar al conjunto una mejora significativa, particularmente en el caso de problemas de tipo local (con importantes variaciones de las fronteras en entornos pequeños). Aunque podemos encontrar trabajos previos en esta línea, éstos no parecen proporcionar los resultados esperados [Wickramaratna et al., 2001, Kim et al., 2003, Rangel et al., 2005, Li et al., 2008, Wu et al., 2009, Lima et al., 2009, Wei et al., 2010]. Este hecho se debe a que el uso

de SVMs como elementos base no puede ser abordado directamente, ya que, a pesar de enfatizar la población de entrenamiento, existe una falta de diversidad entre los aprendices resultantes, lo que ocasiona que su adición directa al conjunto tienda a degradar el rendimiento global [Wickramaratna et al., 2001]. Lo dicho se debe al carácter estable de los aprendices SVM.

Para solventar las limitaciones de los esquemas propuestos con anterioridad, en este capítulo se presenta un nuevo procedimiento que permite entrenar eficientemente clasificadores tipo SVM para su adición a un conjunto RAB [Mayhua-López et al., 2013]. Para ello, se plantea modificar la formulación de la SVM con programación lineal (LPSVM,"Linear Programming SVM") [Vapnik, 1998, Bradley and Mangasarian, 1998, Kecman and Hadzic, 2000, Zhou et al., 2002, Zhu et al., 2004] y combinar esta formulación con un procedimiento de submuestreo adecuado. Dado que la LPSVM regulariza las variables duales de la formulación SVM con un término de penalización L_1, se puede modificar fácilmente para utilizar una matriz núcleos submuestreada por filas, de modo que la LPSVM sólo pueda seleccionar sus Vectores Soporte (SVs) entre un subconjunto de muestras, pero en su formulación siga evaluando el error sobre todos los datos de entrenamiento.

4.1. SVM con programación lineal

El LPSVM es una versión modificada de la formulación SVM donde el factor de regularización cuadrático sobre las variables del primal en la ecuación (1.15), $\|\mathbf{w}\|_2^2$, es sustituido por la regularización de norma uno (1-norm) sobre las variables duales:

$$\begin{aligned} \min_{\mathbf{a},b,\xi} \quad & \|\mathbf{a}\|_1 + C\sum_{l=1}^{L}\xi^{(l)} & \\ \text{s.t.} \quad & y^{(l)}\left[\langle \mathbf{w}, \Phi(\mathbf{x}^{(l)})\rangle + b\right] \geq 1-\xi^{(l)}; & l=1,\ldots,L \\ & \xi^{(l)} \geq 0; & l=1,\ldots,L \end{aligned} \tag{4.1}$$

Seguidamente, las variables del primal, $\mathbf{w}$, se eliminan aplicando el Teorema de Representación ("Representer Theorem") [Kimeldorf and Wahba, 1971], que relaciona $\mathbf{w}$ con $\mathbf{a}$ por medio de

$$\mathbf{w} = \sum_{l'=1}^{L} a^{(l')} \Phi(\mathbf{x}^{(l')}). \tag{4.2}$$

De este modo, la ecuación (4.1) se transforma en

$$\begin{aligned} \underset{\mathbf{a},b,\xi}{\text{mín}} \quad & \|\mathbf{a}\|_1 + C \sum_{l=1}^{L} \xi^{(l)} \\ \text{s.t.} \quad & y^{(l)} \left[\sum_{l'=1}^{L} a^{(l')} K(\mathbf{x}^{(l')}, \mathbf{x}^{(l)}) + b \right] \geq 1 - \xi^{(l)}; \qquad l = 1, \ldots, L \\ & \xi^{(l)} \geq 0; \qquad\qquad\qquad\qquad\qquad\quad l = 1, \ldots, L \end{aligned} \tag{4.3}$$

Resolviendo el problema de optimización (4.3) se obtienen los valores óptimos para los parámetros $\mathbf{a}$ y b. La salida de un clasificador LPSVM para cualquier dato estará dada por

$$f(\mathbf{x}) = \sum_{l'=1}^{L} a^{(l')} K(\mathbf{x}^{(l')}, \mathbf{x}) + b. \tag{4.4}$$

Es importante señalar que el rol del parámetro C en la LPSVM ha cambiado, siendo ahora un compromiso entre el número de errores y el nivel de dispersión en $\mathbf{a}$. Así, se consigue una forma fácil de controlar la complejidad de la función discriminante (4.4).

4.1.1. LPSVM submuestreado como aprendiz de RAB

Con el fin de forzar diversidad en los clasificadores base a la hora de construir el conjunto RAB, se aplica un procedimiento de submuestreo de tal manera que se limite el conjunto de datos que pueden ser SVs. De este modo, los SVs solo podrán ser aquellas muestras que pertenecen al subconjunto indexado por $\mathcal{I}^{L'}$, donde L' $(< L)$ es el número de muestras del subconjunto seleccionado. Este tipo de nuevos aprendices, al cual denominaremos LPSVM submuestreado (SLPSVM), puede entrenarse

resolviendo el siguiente problema de optimización:

$$\begin{aligned} \min_{\mathbf{a},b,\xi} \quad & \sum_{l' \in \mathcal{I}^{L'}} \left| a^{(l')} \right| + C \sum_{l=1}^{L} \xi^{(l)} \\ \text{s.t. :} \quad & y^{(l)} \left[\sum_{l' \in \mathcal{I}^{L'}} a^{(l')} K(\mathbf{x}^{(l')}, \mathbf{x}^{(l)}) + b \right] \geq 1 - \xi^{(l)}; \quad l = 1, \ldots, L \\ & \xi^{(l)} \geq 0; \quad l = 1, \ldots, L \end{aligned} \tag{4.5}$$

Es interesante observar que esta formulación impone que las variables duales, $\mathbf{a}$, solo pueden tener elementos en el subconjunto indexado por $\mathcal{I}^{L'}$, pero las restricciones de clasificación se imponen para todos los datos. El hecho de permitir que cada clasificador considere el error de todas las muestras de entrenamiento evita los problemas de sobreajuste que presentan los métodos propuestos en [Kim et al., 2003, Rangel et al., 2005], proporcionando clasificadores más precisos.

Para añadir un SLPSVM como aprendiz del conjunto RAB, hace falta forzar a este clasificador a que preste atención a las muestras mal clasificadas de acuerdo con la función de énfasis $D_t(l)$. Para ello, en las variables de holgura, $\xi^{(l)}$, del problema de optimización (4.5) se introduce la función de énfasis de RAB, es decir,

$$\begin{aligned} \min_{\mathbf{a}_t,b_t,\xi} \quad & \sum_{l' \in \mathcal{I}_t^{L'}} \left| a_t^{(l')} \right| + C \sum_{l=1}^{L} D_t(l) \xi^{(l)} \\ \text{s.t. :} \quad & y^{(l)} \left[\sum_{l' \in \mathcal{I}_t^{L'}} a_t^{(l')} K(\mathbf{x}^{(l')}, \mathbf{x}^{(l)}) + b_t \right] \geq 1 - \xi^{(l)}; \quad l = 1, \ldots, L \\ & \xi^{(l)} \geq 0; \quad l = 1, \ldots, L \end{aligned} \tag{4.6}$$

donde $\mathcal{I}_t^{L'}$ indica el indice del subconjunto de muestras seleccionadas en la ronda t-ésima de RAB.

Finalmente, desde el punto de vista práctico se deben considerar dos aspectos importantes a la hora del construir el conjunto RAB:

1. En primer lugar, la solución de (4.6) no es directa, ya que la presencia de la función valor absoluto sobre las variables duales no permite el uso de he-

rramientas software de optimización estándar. Afortunadamente, este inconveniente se puede superar fácilmente redefiniendo las variables duales como $a_t^{(l')} = a_{t_+}^{(l')} - a_{t_-}^{(l')}$, con $a_{t_+}^{(l')}, a_{t_-}^{(l')} \geq 0$, tal y como se hace en los algoritmos que maximizan el margen aplicados a problemas de regresión [Schölkopf and Smola, 2002]. Por lo tanto, la norma 1 sobre las variables duales se puede expresar como $\sum_{l' \in S_t^{L'}} |a_t^{(l')}| = \sum_{l' \in S_t^{L'}} (a_{t_+}^{(l')} + a_{t_-}^{(l')})$[1], y (4.6) se convierte en un problema de Programación Lineal (LP, "Linear Programming"):

$$\begin{aligned} \min_{\mathbf{a}_t, b_t, \xi} \quad & \sum_{l' \in S_t^{L'}} \left(a_{t_+}^{(l')} + a_{t_-}^{(l')} \right) + C \sum_{l=1}^{L} D_t(l) \xi^{(l)} & \\ \text{s.t. :} \quad & y^{(l)} \left[\sum_{l' \in S_t^{L'}} \left(a_{t_+}^{(l')} - a_{t_-}^{(l')} \right) K(\mathbf{x}^{(l')}, \mathbf{x}^{(l)}) + b_t \right] \geq 1 - \xi^{(l)}; & l = 1, \ldots, L \\ & \xi^{(l)} \geq 0; & l = 1, \ldots, L \\ & a_{t_+}^{(l')}, \quad a_{t_-}^{(l')} \geq 0; & l' \in S_t^{L'} \end{aligned} \tag{4.7}$$

que puede ser resuelto por cualquier herramienta software de LP. Después de calcular los valores óptimos de $\{a_{t_+}^{(l')}\}_{l'=1}^{L'}$, $\{a_{t_-}^{(l')}\}_{l'=1}^{L'}$ y, b_t, la salida del t-ésimo aprendiz SLPSVM estará dada por:

$$f_t(\mathbf{x}) = \sum_{l' \in S_t^{L'}} \left(a_{t_+}^{(l')} - a_{t_-}^{(l')} \right) K(\mathbf{x}^{(l')}, \mathbf{x}^{(l)}) + b_t \tag{4.8}$$

2. El segundo aspecto práctico es que el rango de valores de la salida (sobre las muestras de entrenamiento) de la SLPSVM tiene que estar contenido en el intervalo $[-1, 1]$ para poder aplicar el algoritmo RAB. Esto se consigue dividiendo

[1] Téngase en cuenta que esta expresión se sostiene porque el problema de optimización está obligando implícitamente a que al menos uno de los términos, $a_{t_+}^{(l')}$ o $a_{t_-}^{(l')}$, sea cero, dependiendo de si el valor óptimo de $a_t^{(l')}$ es positivo ($a_{t_+}^{(l')} > 0$ y $a_{t_-}^{(l')} = 0$), negativo ($a_{t_-}^{(l')} > 0$ y $a_{t_+}^{(l')} = 0$) o cero ($a_{t_+}^{(l')} = a_{t_-}^{(l')}$).

la salida por su valor absoluto máximo sobre las muestras de entrenamiento, es decir, empleando

$$\tilde{f}_t(\mathbf{x}) = \frac{f_t(\mathbf{x})}{\text{máx}_{l=1,\ldots,L} |f_t(\mathbf{x})|} \tag{4.9}$$

en lugar de (4.8).

4.1.2. Procedimiento de submuestreo

A la hora de entrenar los clasificadores base, una selección adecuada de las muestras pertenecientes a los subconjuntos submuestreados influye significativamente en el rendimiento final del conjunto. Cuando se desea submuestrear según el peso asignado a cada muestra por la función de énfasis, son tres las estrategias comunmente utilizadas [Kalal et al., 2008]:

1. **"Trimming"**: esta estrategia selecciona sólo las muestras con mayor peso;
2. **Muestreo Uniforme Único** (UUS, "Unique Uniform Sampling"), donde todas las muestras tienen la misma probabilidad de ser seleccionadas; y
3. **Muestreo Ponderado** (WS, "Weighted Sampling"), que asigna a cada muestra una probabilidad diferente de ser seleccionada (dependiendo del valor del peso asignado por el énfasis) y también permite el reemplazamiento durante el proceso de submuestreo.

Para aplicar una técnica de submuestreo para construir el conjunto RAB-SLPSVM, en primer lugar, se debe tener en cuenta el criterio del énfasis durante el proceso de submuestreo, y, en segundo lugar, se debe tratar de encontrar una solución de compromiso entre la diversidad de los aprendices y el número de SVs compartidos entre SLPSVMs diferentes, de modo que la complejidad del conjunto resultante se reduzca.

El procedimiento de "trimming" no es adecuado para obtener suficiente diversidad entre los aprendices, ya que las muestras más enfatizadas serían siempre las

mismas durante el crecimiento del conjunto, y por tanto no resultaría posible entrenar aprendices diversos. Sin embargo, combinar los enfoques WS y UUS puede ser una buena estrategia, ya que puede proporcionar subconjuntos de entrenamiento heterogéneos con muestras distribuidas en todo el espacio de entrada.

De este modo, si L' es el número de muestras que se desea seleccionar, el procedimiento de submuestreo propuesto funciona de la siguiente manera: en primer lugar se forman L' subconjuntos de muestras ordenadas según la importancia asignada por la función de énfasis a cada muestra (en cada subconjunto las muestras tienen valores similares de énfasis) y, seguidamente, de cada subconjunto se selecciona una muestra según el método UUS. De esta manera, el subconjunto final consta de L' muestras que representan el problema, ya que se garantiza la selección de muestras correctamente y erróneamente clasificadas distribuidas en todo el espacio de entrada.

4.2. RAB con aprendices SLPSVM

Para construir el conjunto que denominamos RAB-SLPSVM, se procede de modo similar al RAB estándar. Durante una serie de rondas, $t = 1, \ldots, T$, se entrena un clasificador SLPSVM que implementa una función $\tilde{f}_t(\mathbf{x}) \in [-1, 1]$, asignándole un peso de salida, α_t, y lo añade al conjunto de modo que la salida global del sistema, $F_T(\mathbf{x})$, se obtenga como combinación lineal ponderada de todos los clasificadores base

$$F_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t \tilde{f}_t(\mathbf{x}). \tag{4.10}$$

Ya se sabe que cuando llegue un nuevo dato la etiqueta asignada por el sistema será

$$\hat{y}(\mathbf{x}) = \operatorname{sgn}\left[F_T(\mathbf{x})\right]. \tag{4.11}$$

El peso asignado a la salida del t-ésimo clasificador base es, como en el RAB estándar,

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1+\gamma_t}{1-\gamma_t}\right), \tag{4.12}$$

donde $\gamma_t = \sum_{l=1}^{L} D_t(\mathbf{x}^{(l)}) \tilde{f}_t(\mathbf{x}^{(l)}) y^{(l)}$ es el parámetro de separación del t-ésimo clasificador y $D_t(\cdot)$ es la función de énfasis. La regla de actualización del énfasis es:

$$D_{t+1}(\mathbf{x}^{(l)}) = \frac{D_t(\mathbf{x}^{(l)}) \exp[-\alpha_t \tilde{f}_t(\mathbf{x}^{(l)}) y^{(l)}]}{Z_t}, \tag{4.13}$$

donde Z_t es un factor de normalización, que se calcula mediante

$$Z_t = \sum_{l=1}^{L} D_t(\mathbf{x}^{(l)}) \exp[-\alpha_t \tilde{f}_t(\mathbf{x}^{(l)}) y^{(l)}] \tag{4.14}$$

para asegurar que $\sum_{l=1}^{L} D_{t+1}(\mathbf{x}^{(l)}) = 1$.

En la Tabla 4.2 se recoge el pseudocódigo del algoritmo RAB-SLPSVM.

4.3. Pruebas experimentales

Las prestaciones del algoritmo propuesto, RAB-SLPSVM, se analizan en comparación con los resultados proporcionados por los algoritmos SVM y LPSVM cuando estos se utilizan como clasificadores individuales, y con dos métodos de referencia de Boosting con SVM. Por un lado, el "Diverse AdaBoostSVM" (D-ABSVM) propuesto en [Li et al., 2008] que utiliza SVMs con núcleo Gaussiano y varía la dispersión del núcleo a lo largo del crecimiento del conjunto con el fin de forzar diversidad entre los aprendices. Por otro lado, el "Weakened SVM" (AB-WSV) propuesto en [Rangel et al., 2005], que limita la capacidad expresiva de los aprendices al entrenar cada SVM con un subconjunto reducido de muestras seleccionadas según una constante de debilidad propia de este método.

4.3.1. Bases de datos

Para la evaluación de los métodos propuestos se han considerado doce bases de datos de diferentes características (tamaño, dimensión y dificultad). De las ocho bases de datos utilizados en el capítulo 3, los problemas Cr y He no se consideren

Tabla 4.1: Pseudocódigo del algoritmo RAB-SLPSVM.

1 - Entradas: $\{\mathbf{x}^{(l)}, y^{(l)}\}_{l=1}^{L}$.

2 - Inicialización: $D_1(\mathbf{x}^{(l)}) = 1/L \quad \forall l$.

3 - Para $t = 1, \ldots, T$

3.1 - Entrenar un clasificador SLPSVM, $f_t(\mathbf{x}^{(l)})$, y limitar su salida al intervalo $[-1, 1]$:

$$\tilde{f}_t(\mathbf{x}) = \frac{f_t(\mathbf{x})}{\text{máx}_{l=1,\ldots,L}|f_t(\mathbf{x})|}$$

3.2 - Calcular el peso de salida asociado a este clasificador como

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1+\gamma_t}{1-\gamma_t}\right)$$

donde $\gamma_t = \sum_{l=1}^{L} D_t(\mathbf{x}^{(l)})\tilde{f}_t(\mathbf{x}^{(l)})y^{(l)}$.

3.3 - Actualizar la función de énfasis para la siguiente iteración:

$$D_{t+1}(\mathbf{x}^{(l)}) = \frac{D_t(\mathbf{x}^{(l)})\exp[-\alpha_t\tilde{f}_t(\mathbf{x}^{(l)})y^{(l)}]}{Z_t}$$

siendo $Z_t = \sum_{l=1}^{L} D_t(\mathbf{x}^{(l)})\exp[-\alpha_t\tilde{f}_t(\mathbf{x}^{(l)})y^{(l)}]$.

4 - El clasificador final implementa la función:

$$F_T(\mathbf{x}) = \sum_{t=1}^{T}\alpha_t\tilde{f}_t(\mathbf{x})$$

siendo, por tanto, $\hat{y}(\mathbf{x}) = \text{sign}\,[F_T(\mathbf{x})]$ la clase estimada para el patrón $\mathbf{x}$.

por poseer pocas muestras de entrenamiento. Las restantes seis se han obtenido del repositorio de la UCI [Frank and Asuncion, 2010] (Diabetes, Duke breast-cancer, Madelon, Thyroid, Twonorm, y Waveform.).

En la Tabla 4.2 se resumen las principales características de cada uno de estos problemas (d: dimensión; L_1/L_{-1}: número de muestras) para los conjuntos de entrenamiento y test. La notación o abreviatura con la que se va a denotar cada problema

es con sus dos primeras letras. En el Apéndice C se presenta una descripción más detallada de cada uno de estos problemas.

Tabla 4.2: Principales características de las bases de datos.

Problemas	**Dimensión** d	**Nº. Muestras entrenamiento** L_1/L_{-1}	**Nº. Muestras test** L_1/L_{-1}
Abalone (Ab)	8	1238/1269	843/827
Breast (Br)	9	145/275	96/183
Contraceptive (Co)	9	506/377	338/252
Diabetes (Di)	8	172/296	96/204
Duke breast-cancer (Du)	7128	20/23	25/18
Ionosfera (Io)	34	101/100	124/26
Kwok (Kw)	2	300/200	6120/4080
Madelon (Ma)	500	1000/1000	300/300
Ripley (Ri)	2	125/125	500/500
Thyroid (Th)	5	43/97	22/53
Twonorm (Tw)	20	199/201	3504/3496
Waveform (Wa)	21	124/276	1523/3077

4.3.2. Entrenamiento de los conjuntos a evaluar

El entrenamiento del algoritmo RAB-SLPSVM se realiza según indica el pseudocódigo recogido en la Tabla 4.1; los conjuntos D-ABSVM y AB-WSV, siguiendo los procedimientos descritos en [Li et al., 2008] y [Rangel et al., 2005], respectivamente.

Todos los algoritmos han sido implementados en MATLAB R2007b y los problemas de optimización se han resuelto con la herramienta de optimización MOSEK [Mosek, 2010].

Diseño de los clasificadores base

Los clasificadores tipo SVMs utilizan una función Gaussiana como núcleo; por lo tanto, los parámetros libres a ajustar son el valor de C y la dispersión del núcleo, σ.

En RAB-SLPSVM los aprendices son del tipo SLPSVM (descrito en el Apartado 4.1.1). Este tipo de aprendices emplea un parámetro adicional, el número de muestras de los subconjuntos submuestreadas, L'.

El algoritmo AB-WSV entrena los aprendices SVMs con la formulación ν-SVM [Schölkopf and Smola, 2002], donde el parámetro de regularización C se reemplaza por la constante ν. Adicionalmente, para debilitar los aprendices hace uso de un parámetro de debilidad γ que permite calcular el número de muestras del subconjunto correspondiente a la t-ésima ronda de Boosting.

Para el método restante, D-ABSVM, el valor de la dispersión del núcleo Gaussiano cambia según crece el conjunto desde un valor inicial, σ_{ini}, hasta un valor mínimo, σ_{min}, en pasos de σ_{step}. Además, en cada paso se compara la diversidad del t-ésimo clasificador con un umbral de diversidad κ. Si la diversidad del t-ésimo clasificador supera el umbral κ, el clasificador se suma al conjunto; de lo contrario, se rechaza y se vuelve a entrenar un nuevo clasificador.

Selección del número de clasificadores base

Para detener el crecimiento del conjunto se hace uso del procedimiento descrito en el Apartado 3.4.2, que atiende a la evolución de los valores de α_t de acuerdo con el criterio propuesto en [Gómez-Verdejo et al., 2008]. Concretamente, dicho criterio consiste en detener el crecimiento del conjunto cuando se cumple que

$$\frac{\sum_{t=T-9}^{T} \alpha_t}{\sum_{t=1}^{T} \alpha_t} < T_{stop} \tag{4.15}$$

donde T_{stop} se fija de manera empírica a 0.25 para todos los problemas y algoritmos excepto para el problema TW y el algoritmo AB-WSV, donde el valor de T_{stop} se fija en 0.35.

Selección de los parámetros de diseño

Los parámetros de diseño se han seleccionado mediante el proceso de Validación Cruzada (CV) de 5 particiones en el que se han empleado 10 repeticiones sobre cada partición.

Los valores de C y σ se exploran en 12 valores logaritmicamente espaciados en los intervalos $[10^{-2}, 10^{5}]$ y $\left[10^{-1}\sqrt{d}, 10^{2}\sqrt{d}\right]$, respectivamente.

El parámetro adicional para RAB-SLPSVM, L', se explora en siete valores: 5 %, 10 %, 15 %, 25 %, 50 %, 75 % y 100 % de las L muestras disponibles.

Para el caso de AB-WSV, los parámetros adicionales a validar son ν y γ. Ambos se exploran en nueve valores linealmente espaciados en el intervalo [0.1, 0.9]. Cabe recordar que, en este método, los aprendices se entrenan con la formulación ν-SVM; por lo tanto, el parámetro C no interviene.

El ajuste de los parámetros del algoritmo D-ABSVM no es sencillo, y una selección aplicando el procedimiento de CV tiende a proporcionar conjuntos con rendimiento muy pobre. Por este motivo, los valores se han establecido según recomiendan sus autores en [Li et al., 2008], es decir: el parámetro C se fija en 50 por motivos empíricos. El valor de σ_{ini} se define como el radio de dispersión de las muestras en el espacio de entrada; σ_{min} se calcula como el promedio de la distancia mínima entre muestras; σ_{step} se establece en 2, tambien por motivos empiricos; y el factor de diversidad, κ, se fija en 0.7.

4.3.3. Resultados

En la Tabla 4.3 se muestran las tasas de error de clasificación en test, CE, el tamaño de los conjuntos, T, y el número de SVs, N_{sv}, obtenidos después entrenar los conjuntos usando los valores de los parámetros seleccionados por CV. Para el caso de RAB-SLPSVM, los resultados mostrados son los obtenidos después de promediar sobre 50 repeticiones.

Respecto al número de SVs, durante el crecimiento del conjunto es posible que

Tabla 4.3: Tasas de error de clasificación, CE, tamaños de los conjuntos, T, y números de SVs, N_{sv}, correspondientes a los algoritmos SVM, LPSVM, D-ABSVM, AB-WSV y RAB-SLPSVM.

		SVM	**LPSVM**	**D-ABSVM**	**AB-WSV**	**RAB-SLPSVM**
Ab	$CE(\%)$	20.1	20.0	22.4	19.4	**19.1** ± 0.1
	T	–	–	51.0	3.0	13.9 ± 1.8
	N_{SV}	1175	59.0	2507	1374	16.0 ± 3.0
Br	$CE(\%)$	2.5	2.5	2.5	4.3	**2.2** ± 0.2
	T	–	–	69.0	7.0	10.1 ± 0.8
	N_{SV}	50.0	4.0	420	278	59.0 ± 5.0
Co	$CE(\%)$	28.3	28.6	33.9	35.3	**27.9** ± 0.1
	T	–	–	65.0	9.0	15.1 ± 0.5
	N_{SV}	578	27.0	883	710	265 ± 48.0
Di	$CE(\%)$	19.3	22.3	20.7	20.3	**18.2** ± 0.3
	T	–	–	95.0	7.0	21.0 ± 0.0
	N_{SV}	264	31.0	468	275	94.0 ± 6.0
Du	$CE(\%)$	13.9	14.0	15.3	14.0	**12.0** ± 1.2
	T	–	–	73.0	4.0	17.3 ± 0.6
	N_{SV}	43.0	19.0	43.0	36.0	36.0 ± 3.0
Io	$CE(\%)$	**2.0**	**2.0**	4.7	9.3	**2.0** ± 0.7
	T	–	–	191	8.0	23.5 ± 4.0
	N_{SV}	121	26.0	201	104	170 ± 7.0
Kw	$CE(\%)$	11.8	11.7	13.1	12.2	**11.5** ± 0.0
	T	–	–	10.0	2.0	16.9 ± 0.6
	N_{SV}	132	14.0	500	447	129 ± 0.7
Ma	$CE(\%)$	42.2	43.5	44.2	41.8	**40.8** ± 0.6
	T	–	–	73.0	14.0	9.0 ± 0.8
	N_{SV}	186	77.0	1977	1249	179 ± 10
Ri	$CE(\%)$	9.5	9.4	10.8	9.2	**8.8** ± 0.2
	T	–	–	22.0	8.0	13.9 ± 0.7
	N_{SV}	91.0	16.0	250	183	88.0 ± 6.0
Th	$CE(\%)$	5.3	6.7	12.0	5.3	**4.5** ± 0.8
	T	–	–	91.0	2.0	14.8 ± 2.3
	N_{SV}	85.0	20.0	140	100	65.0 ± 9.0
Tw	$CE(\%)$	2.5	2.8	**2.4**	3.2	2.5 ± 0.1
	T	–	–	2.0	3.0	6.6 ± 0.5
	N_{SV}	117	12.0	400	61.0	51.0 ± 4.0
Wa	$CE(\%)$	10.5	11.2	**10.4**	13.1	10.5 ± 0.1
	T	–	–	99.0	6.0	25.0 ± 0.1
	N_{SV}	121	16.0	400	271	136 ± 7.0

un mismo dato se repita como SVs en dos o más aprendices que pertenecen a rondas diferentes; en estos casos, el SVs se cuenta una sola vez, ya que la salida del conjunto final se puede calcular evaluando este núcleo una sola vez (agrupando).

Para facilitar la comparación entre algoritmos, se resalta en negrita el mejor resultado (el que presenta el menor error de clasificación).

Analizando los resultados mostrados en la Tabla 4.3, se puede concluir que, en la mayoría de casos, el algoritmo RAB-SLPSVM consigue mejores resultados, con las dos únicas excepciones de los problemas Tw y Wa, donde D-ABSVM consigue una ligera ventaja. Un análisis más detallado nos dice que:

- En todos los casos, RAB-SLPSVM supera ampliamente los resultados obtenidos con el algoritmo AB-WSV. Si bien es cierto que AB-WSV emplea un menor número de aprendices, también incluye un número elevado de datos como SVs. Este comportamiento hace que el conjunto resultante tenga una estructura de mayor tamaño; por lo tanto, de mayor coste computacional al momento de clasificar un nuevo dato.

- El algoritmo D-ABSVM conduce sistemáticamente a estructuras de mayor tamaño. De hecho, en los dos únicos problemas (Tw y Wa) donde D-ABSVM consigue ligeras ventajas, el conjunto incluye todas la muestras de entrenamiento como SVs. Este comportamiento puede estar relacionado con el carácter sintético de los problemas Tw y Wa, ya que existe cierta similitud entre los conjuntos de entrenamiento y test.

- Si se compara el algoritmo RAB-SLPSVM con las versiones individuales SVM y LPSVM, en todos los casos se consigue mejorar las tasas de error, salvo tres empates (Io, Tw y Wa) con el SVM y un único empate (Io) con LPSVM. Incluso, en la mayoría de casos, RAB-SLPSVM supone un menor número de muestras como SVs que una SVM individual. Obviamente, LPSVM consigue soluciones más dispersas.

Con el fin de analizar si las ventajas del método propuesto se deben a los aprendices LPSVM, al proceso de submuestreo, o a su adecuada combinación, se implementan versiones de Boosting intermedias donde se emplean clasificadores SVM y LPSVM estándar como aprendices considerando el énfasis de RAB. Las dos primeras versiones, que denominamos como RAB-SVM o RAB-LPSVM según sea el tipo de clasificador, son versiones sin aplicar submuestreo. La tercera, que emplea como aprendices SVMs submuestreadas, será denominada RAB-SSVM. Todos los conjuntos se entrenan siguiendo el procedimiento mostrado en la Tabla 4.1, con el mismo criterio descrito en el Apartado 4.3.2. Los resultados obtenidos se muestran en la Tabla 4.4.

A la vista de los resultados mostrados en la Tabla 4.4, podemos extraer las siguientes conclusiones:

- Las ventajas que se obtienen con el método RAB-SLPSVM propuesto se confirman: las versiones intermedias no consiguen mejoras. RAB-SVM consigue resultados similares para cuatro de los casos: Br, Io, Ri y Wa. Sin embargo, mejorar estos resultados es complicado, ya que están muy cerca de los mínimos registrados (el limite teórico para Ri=8.3 % [Ripley, 1994] y los mínimos registrados empíricamente para Br=2.1 % y Io=1.8 %, según [Frank and Asuncion, 2010]). En el problema Wa, hay similitud entre los conjuntos de entrenamiento y test por el carácter sintético de la base de datos.

- En todo caso, se puede notar que emplear clasificadores tipo SVM (ya sean SVMs o LPSVMs) de forma directa no otorga ventajas; incluso en ocasiones degrada las prestaciones. Comportamiento que también se observó en [Wickramaratna et al., 2001]. Sin embargo, RAB-LPSVM consigue claramente emplear un menor número de datos como SVs.

- Si nos fijamos en los resultados de la versión submuestreada, RAB-SSVM, tampoco se observan mejoras. Es decir, aplicar un procedimiento de submuestro de forma directa no otorga ventajas; por el contrario, en algunos casos lleva a

Tabla 4.4: Prestaciones presentadas por los algoritmos RAB-SVM, RAB-SSVM, RAB-LPSVM y RAB-SLPSVM.

		RAB-SVM	**RAB-SSVM**	**RAB-LPSVM**	**RAB-SLPSVM**
Ab	$CE(\%)$	21.4	20.9 ± 0.4	21.7	**19.1** ± 0.1
	T	12.0	8.5 ± 1.0	18.0	13.9 ± 1.8
	N_{SV}	1835	1254 ± 10	18.0	16.0 ± 3.0
Br	$CE(\%)$	**2.2**	3.2 ± 0.5	**2.2**	**2.2** ± 0.2
	T	8.0	9.6 ± 2.4	10.0	10.1 ± 0.8
	N_{SV}	131	209 ± 0.0	17.0	59.0 ± 5.0
Co	$CE(\%)$	30.2	30.5 ± 1.4	28.8	**27.9** ± 0.1
	T	15.0	12.0 ± 1.3	14.0	15.1 ± 0.5
	N_{SV}	748	221 ± 3.0	42.0	265 ± 48.0
Di	$CE(\%)$	20.7	22.3 ± 1.4	26.0	**18.2** ± 0.3
	T	9.0	7.9 ± 1.4	16.0	21.0 ± 0.0
	N_{SV}	387	234 ± 0.0	194	94.0 ± 6.0
Du	$CE(\%)$	14.5	14.1 ± 1.1	14.2	**12.0** ± 1.2
	T	16.0	17.8 ± 1.2	19.0	17.3 ± 0.6
	N_{SV}	43.0	43.0 ± 0.0	39.0	36.0 ± 3.0
Io	$CE(\%)$	**2.0**	3.5 ± 1.2	2.7	**2.0** ± 0.7
	T	18.0	11.4 ± 1.0	18.0	23.5 ± 4.0
	N_{SV}	200	101 ± 0.0	38.0	170 ± 7.0
Kw	$CE(\%)$	11.8	13.2 ± 1.1	11.8	**11.5** ± 0.0
	T	13.0	7.6 ± 1.4	15.0	16.9 ± 0.6
	N_{SV}	500	50.0 ± 0.0	20.0	129 ± 0.7
Ma	$CE(\%)$	43.6	42.3 ± 0.9	44.7	**40.8** ± 0.6
	T	20.0	17.1 ± 1.2	21.0	9.0 ± 0.8
	N_{SV}	934	830 ± 10	540	179 ± 10
Ri	$CE(\%)$	**8.8**	11.8 ± 2.4	9.2	**8.8** ± 0.2
	T	7.0	9.6 ± 2.1	12.0	13.9 ± 0.7
	N_{SV}	233	25.0 ± 0.0	44.0	88.0 ± 6.0
Th	$CE(\%)$	5.0	4.7 ± 1.1	6.7	**4.5** ± 0.8
	T	20.0	22.2 ± 1.6	13.0	14.8 ± 2.3
	N_{SV}	91.0	35.0 ± 0.0	19.0	65.0 ± 9.0
Tw	$CE(\%)$	2.5	4.3 ± 1.4	3.5	2.5 ± 0.1
	T	7.0	18.9 ± 2.9	11.0	6.6 ± 0.5
	N_{SV}	366	60.0 ± 0.0	11.0	51.0 ± 4.0
Wa	$CE(\%)$	**10.4**	11.2 ± 0.2	11.3	10.5 ± 0.1
	T	14.0	14.5 ± 0.6	16.0	25.0 ± 0.1
	N_{SV}	184	100 ± 0.0	21.0	136 ± 7.0

estructuras con mayor número de elementos y de peores prestaciones.

Según lo analizado, las ventajas del algoritmo RAB-SLPSVM se deben a una (adecuada) combinación de dos modificaciones del RAB: el uso de aprendices LPSVM, que proporcionan soluciones dispersas, y un adecuado procedimiento de submuestreo, que permite trabajar con matrices núcleo submuestreadas por filas. La combinación permite, por un lado controlar la capacidad expresiva de los aprendices, y por otro garantiza la diversidad entre los elementos del conjunto.

4.3.4. Análisis de la sensibilidad respecto al submuestreo

Como se ha mencionado en la sección anterior, una de las ventajas del algoritmo RAB-SLPSVM es que permite trabajar con matrices núcleo submuestreadas por filas. En esta sección se va ha llevar a cabo una evaluación de la sensibilidad del algoritmo al parámetro adicional L'. Para ello, en la Tabla 4.5 se muestran las tasas de error de clasificación (promediadas sobre 50 repeticiones), CE, y los tamaños de los conjuntos, T, para diferentes valores de L'.

De los resultados mostrados en la Tabla 4.5 se pueden sacar las siguientes conclusiones sobre la influencia de L':

- En la mayoría de casos (la única excepción es Du), cuando L' toma valores pequeños (alrededor de 5 %) o valores superiores al 50 %, las prestaciones se degradan.

- El valor de L' que tiende a proveer mejores resultados está alrededor de 15 %. Los casos donde el valor de L' se aleja del 15 % son Du, Ri y Tw. Para los problemas Ri y Tw, las diferencias no son apreciables; pero sí en Du. Eso se debe a que Du es un problema con pocas muestras de entrenamiento y de alta dimensión.

- Respecto al tamaño de los conjuntos, se observa un comportamiento similar. En

Tabla 4.5: Tasas de error de clasificación, CE, y tamaños de los conjuntos, T, con diferentes valores de L', proporcionados por el algoritmo RAB-SLPSVM.

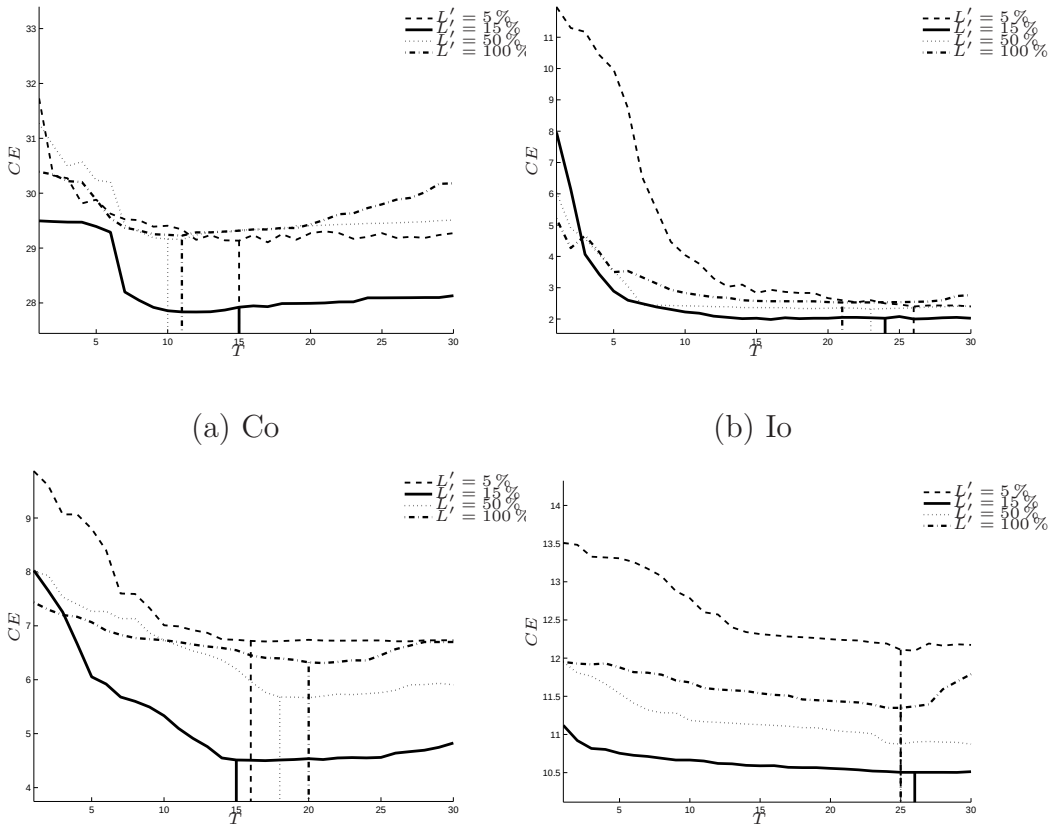
		5 %	10 %	15 %	25 %	50 %	75 %	100 %
	CE	21.9 ± 0.3	20.9 ± 0.2	**19.1** ± 0.1	19.5 ± 0.1	21.2 ± 0.1	21.5 ± 0.1	22.1 ± 0.1
Ab	T	20.3 ± 0.9	18.3 ± 0.6	13.9 ± 0.1	17.1 ± 0.8	20.1 ± 0.3	21.3 ± 0.4	20.2 ± 0.3
	CE	2.4 ± 0.3	**2.2** ± 0.2	2.2 ± 0.2	2.3 ± 0.2	2.4 ± 0.2	2.4 ± 0.1	2.5 ± 0.1
Br	T	20.0 ± 0.1	10.1 ± 0.8	14.3 ± 0.7	14.2 ± 0.9	16.3 ± 0.9	14.5 ± 0.7	15.2 ± 0.9
	CE	29.1 ± 0.4	28.7 ± 0.4	**27.9** ± 0.1	28.8 ± 0.3	29.1 ± 0.1	29.2 ± 0.1	29.2 ± 0.1
Co	T	15.1 ± 1.2	10.3 ± 1.4	15.1 ± 0.5	13.5 ± 1.0	10.2 ± 1.5	11.2 ± 1.3	10.1 ± 1.5
	CE	19.3 ± 0.5	18.5 ± 0.4	**18.2** ± 0.3	19.1 ± 0.6	19.5 ± 0.2	19.7 ± 0.2	20.3 ± 0.2
Di	T	20.0 ± 0.7	25.1 ± 1.3	21.0 ± 0.0	20.1 ± 0.1	22.3 ± 0.2	21.3 ± 0.1	37.1 ± 1.7
	CE	17.3 ± 1.2	17.2 ± 1.2	15.3 ± 1.3	14.2 ± 1.0	**12.0** ± 1.2	13.3 ± 1.2	16.9 ± 1.4
Du	T	25.2 ± 0.7	30.5 ± 0.3	29.1 ± 0.9	15.0 ± 1.2	17.3 ± 0.6	18.5 ± 1.7	19.3 ± 1.2
	CE	2.3 ± 0.9	2.2 ± 0.9	**2.0** ± 0.7	2.1 ± 0.7	2.3 ± 0.8	2.5 ± 0.9	2.7 ± 0.9
Io	T	26.3 ± 4.5	22.1 ± 3.7	23.5 ± 4.0	22.6 ± 3.0	23.0 ± 2.5	22.5 ± 2.3	21.1 ± 3.2
	CE	11.9 ± 0.1	11.7 ± 0.0	**11.5** ± 0.0	11.7 ± 0.0	11.8 ± 0.0	11.8 ± 0.0	11.9 ± 0.0
Kw	T	17.1 ± 0.9	18.2 ± 0.5	16.9 ± 0.6	20.0 ± 0.5	19.3 ± 0.9	21.4 ± 0.5	20.7 ± 0.3
	CE	45.3 ± 0.7	42.4 ± 0.8	**40.8** ± 0.6	41.5 ± 0.7	43.2 ± 0.3	44.5 ± 0.2	44.9 ± 0.1
Ma	T	12.3 ± 0.9	14.5 ± 0.7	9.0 ± 0.8	17.3 ± 0.2	10.3 ± 0.3	12.4 ± 0.1	12.1 ± 0.3
	CE	9.5 ± 0.4	9.2 ± 0.2	8.9 ± 0.2	**8.8** ± 0.2	9.3 ± 0.1	9.5 ± 0.1	9.7 ± 0.1
Ri	T	16.2 ± 0.9	15.3 ± 0.5	14.3 ± 0.85	13.9 ± 0.7	18.0 ± 0.9	19.3 ± 0.5	19.0 ± 0.7
	CE	6.7 ± 1.2	5.1 ± 1.2	**4.5** ± 0.8	6.2 ± 1.0	5.7 ± 0.8	5.9 ± 0.8	6.3 ± 0.9
Th	T	16.3 ± 2.9	15.4 ± 2.0	14.8 ± 2.3	17.3 ± 2.5	18.4 ± 2.2	17.3 ± 2.1	20.1 ± 4.2
	CE	2.7 ± 0.4	**2.5** ± 0.1	2.6 ± 0.1	2.6 ± 0.1	2.7 ± 0.1	3.0 ± 0.1	3.2 ± 0.1
Tw	T	7.9 ± 0.3	6.6 ± 0.5	8.3 ± 0.9	7.5 ± 0.3	8.2 ± 0.2	9.0 ± 0.7	9.3 ± 0.2
	CE	12.1 ± 0.4	10.9 ± 0.1	**10.5** ± 0.1	10.7 ± 0.1	10.9 ± 0.1	11.0 ± 0.1	11.3 ± 0.1
Wa	T	25.0 ± 0.2	26.3 ± 0.4	25.0 ± 0.1	23.4 ± 0.2	24.5 ± 0.5	26.2 ± 0.5	25.2 ± 0.9

la mayoría de casos, el valor de L' que consigue conjuntos con menor número de aprendices esta alrededor del 15 %.

En conclusión, si L' se prefija de manera empírica a 15 %, las prestaciones del algoritmo no sufren ninguna degradación significativa; en cambio, se logra disminuir el esfuerzo computacional en la fase de diseño.

En la Figura 4.1 se presentan las curvas de evolución del error de clasificación en función del número de aprendices para cuatro problemas representativos entrenados con el algoritmo propuesto RAB-SLPSVM. Se eligen cuatro valores de L': 5 %, 15 %, 50 % y 100 %, con el fin de visualizar el comportamiento tanto en los extremos como

en las zonas donde se consiguen los mejores resultados.



(a) Co (b) Io

(c) Th (d) Wa

Figura 4.1: Curvas de evolución del error de clasificación CE en función del número de rondas, T, y diferentes valores de L' para cuatro problemas representativos. También se muestran los puntos de parada

Las curvas del la Figura 4.1 evidencian una posible falta de diversidad entre los aprendices cuando L' toma valores en los extremos. El valor de CE no consigue descender del todo: llegado a un punto se estanca, y comienza a repuntar cuando L' es el 100 % de L.

Cuando L' toma el valor de 5 %, la evolución de CE tiene un comportamiento normal para este tipo de esquemas, con una ligera tendencia al sobreajuste. Esto

se debe, probablemente, a la potencia expresiva de los aprendices; en consecuencia, es necesario vigilar el sobreajuste durante el proceso de diseño del conjunto RAB-SLPSVM.

4.3.5. Carga computacional en entrenamiento y operación

Dado que los elementos del conjunto RAB-SLPSVM son SVMs dispersas, es necesario realizar un análisis sobre la carga computacional, tanto en entrenamiento como en operación, para probar si también se obtienen ventajas en las fases de diseño y operación.

Aunque el número de núcleos de cada clasificador es un buen indicador sobre la carga computacional en operación, medir el tiempo de entrenamiento y clasificación arrojaría mayor información para una comparación con mayor detalle.

Para ello, los algoritmos SVM, LPSVM, D-ABSVM, AB-WSV y RAB-SLPSVM se entrenan (con los valores de los parámetros no entrenables obtenidos según se indica en el Apartado 4.3.2) y se prueban en un ordenador con un procesador Intel (R) Xeon (R) E540 con 3.0 GHz de velocidad de procesamiento y 8.0 GB de memoria RAM. Todos los métodos se implementan con Matlab utilizando el software de optimización MOSEK [Mosek, 2010].

En la Tabla 4.6 se muestran los resultados obtenidos para el tiempo de entrenamiento (t_r) y el de operación (t_o) en milisegundos.

Como se esperaba, el tiempo de entrenamiento para LPSVM es menor que los métodos basados en SVMs, lo que se debe a que la programación lineal es más rápida que la programación cuadrática. Respecto al algoritmo propuesto, RAB-SLPSVM, el tiempo de entrenamiento aumenta linealmente con el número de aprendices, pero en ningún caso es mayor que los tiempos empleados por los otros conjuntos basados en SVMs.

Con respecto al tiempo de operación, los resultados van de acuerdo con el número de SVs (N_{sv}) y el número de aprendices (T). De nuevo, el algoritmo LPSVM es el

Tabla 4.6: Tiempos de entrenamiento (t_r) y operación (t_o) en milisegundos para los algoritmos SVM, LPSVM, D-ABSVM, AB-WSV y RAB-SLPSVM.

	Tiempo (ms)	SVM	LPSVM	D-ABSVM	AB-WSV	RAB-SLPSVM
Ab	t_o	48 ± 1.0	0.6 ± 0.2	4035 ± 89	5375 ± 123	11 ± 1.8
	t_r	510 ± 12	440 ± 18	10589 ± 604	94344 ± 880	8356 ± 176
Br	t_o	17 ± 0.3	0.2 ± 0.0	30 ± 2.3	96 ± 16	4.0 ± 0.4
	t_r	292 ± 17	157 ± 70	1072 ± 14	16113 ± 87	900 ± 33
Co	t_o	5.9 ± 2.2	0.5 ± 0.3	323 ± 15	1691 ± 57	13 ± 0.8
	t_r	172 ± 25	140 ± 19	70050 ± 217	13231 ± 442	6195 ± 156
Di	t_o	0.9 ± 0.1	0.2 ± 0.0	134 ± 11	156 ± 18	9.0 ± 0.5
	t_r	298 ± 14	239 ± 60	20259 ± 430	10191 ± 44	2918 ± 105
Du	t_o	2.9 ± 0.0	0.7 ± 0.1	30 ± 6.0	100 ± 21	181 ± 12
	t_r	17 ± 1.0	12 ± 1.0	1245 ± 207	1912 ± 92	709 ± 27
Io	t_o	0.2 ± 0.0	0.1 ± 0.0	9.0 ± 0.8	17 ± 2.7	8.0 ± 1.5
	t_r	46 ± 1.0	25 ± 2.0	4357 ± 152	565 ± 61	481 ± 37
Kw	t_o	22 ± 0.9	0.4 ± 0.0	1320 ± 28	1311 ± 71	149 ± 8.3
	t_r	367 ± 7.0	243 ± 34	3904 ± 85	10768 ± 108	2484 ± 39
Ma	t_o	90 ± 4.0	40 ± 4.0	1403 ± 30	1422 ± 60	65 ± 2.4
	t_r	2034 ± 183	1632 ± 171	42032 ± 90	10973 ± 120	3305 ± 190
Ri	t_o	0.6 ± 0.1	0.1 ± 0.0	40 ± 3.8	249 ± 12	13 ± 1.1
	t_r	71 ± 7.0	30 ± 5.0	1091 ± 26	5482 ± 41	1014 ± 106
Th	t_o	0.2 ± 0.0	0.1 ± 0.0	8.8 ± 1.6	44 ± 5.5	4.0 ± 0.1
	t_r	56 ± 3.0	79 ± 4.0	1218 ± 61	1193 ± 43	257 ± 7.0
Tw	t_o	2.8 ± 0.4	2.5 ± 0.6	206 ± 14	842 ± 50	35 ± 1.8
	t_r	222 ± 9.0	211 ± 46	7460 ± 180	10162 ± 106	459 ± 23
Wa	t_o	5.2 ± 0.9	2.1 ± 0.6	6940 ± 84	2660 ± 74	149 ± 5.5
	t_r	227 ± 9.0	198 ± 39	2121 ± 228	11889 ± 87	2715 ± 159

más rápido, dado que contiene un menor número de SVs. El tiempo de operación de RAB-SLPSVM crece con el número de aprendices, pero sigue siendo menor que los empleados por D-ABSVM y AB-WSV, y no mucho mayor y ocasionalmente menor que el SVM estándar.

Finalmente, es importante señalar que estos tiempos son en milisegundos, por lo que el entrenamiento de cualquiera de estos métodos es accesible, y las diferencias entre métodos no son inaceptables.

4.4. Conclusiones

En este capítulo se ha presentado un nuevo procedimiento para utilizar con éxito SVMs como aprendices en Boosting. El método propuesto aplica un procedimiento de submuestreo en la formulación de SVMs con LP, permitiendo emplear una matriz de núcleos submuestreada por filas. Esta combinación de LP y adecuado submuestreo otorga flexibilidad para controlar la diversidad y la complejidad de los aprendices tipo SVM para construir conjuntos por Boosting.

Los experimentos llevados a cabo con doce bases de datos de diferentes características evidencian las buenas prestaciones que ofrece este método. Las ventajas obtenidas no sólo se refieren a una reducción en la tasa del error de clasificación, sino también a la complejidad del conjunto resultante. Se consiguen conjuntos con un reducido número de núcleos, incluso menor que los de una SVM estándar.

En lo relativo a las cargas computacionales precisas para entrenamiento y operación, resultan ser, lógicamente, mayores que las precisas para LPSVM y SVM (en este último caso, la carga de operación puede ocasionalmente llega a ser menor), pero son menores que las correspondientes a los algoritmos D-ABSVM y AB-WSV; en varios casos, muy apreciablemente menores.

Capítulo 5

Conclusiones y líneas de investigación abiertas

En este último capítulo, como es tradicional, se resumen y comentan las principales aportaciones de la presente Tesis Doctoral, y se listan y describen someramente algunas líneas de investigación que abre el trabajo realizado.

5.1. Aportaciones

Entre los conjuntos de máquinas de aprendizaje, los basados en conceptos tipo Boosting presentan particular relevancia, dado que se fundamentan en principios de cooperación–competición. No resulta extraño que se considere que en ello radica la causa de sus muy destacadas prestaciones.

Las distintas versiones de los algoritmos Boosting —y en particular el Real AdaBoost, sobre el que se han apoyado las aportaciones de esta Tesis— no quedan completamente exentas, desde luego, de ciertas limitaciones. De especial importan-

cia resultan las relativas a la restricción de capacidad expresiva que supone utilizar únicamente combinaciones lineales para agregar las salidas de las unidades, así como las referentes a la necesidad de emplear aprendices no sólo (relativamente) débiles, sino inestables (de modo tal que el efecto atencional que provoca el énfasis en cada paso de las muestras más problemáticas hasta llegar a él fuerce la debida diversidad). A aliviar estas dos limitaciones se han dirigido los trabajos realizados.

Primeramente, se ha comprobado que la inclusión de puertas —al modo de las empleadas en Mezclas de Expertos— en el proceso de fusión de salidas de aprendices, gobernando el entrenamiento de dichas puertas mediante la optimización del coste habitual para los conjuntos Real AdaBoost a fin de evitar que se pierda la aconsejable debilidad de las unidades, constituye un mecanismo eficaz para combatir la primera de las limitaciones antes mencionadas. Naturalmente, hay un coste a pagar: un sensible incremento del esfuerzo computacional necesario para entrenar el conjunto, debido a que hay que incluir en la Validación Cruzada los parámetros correspondientes a las puertas. En todo caso, los resultados presentados en el capítulo 3 para las formas particulares de diseño que se han considerado muestran que es posible decir que, en buena parte de los casos, este esfuerzo adicional queda recompensado por unas mejores prestaciones. Y no sólo eso: también frecuentemente se obtienen diseños cuya carga computacional en operación —clasificación de una nueva muestra— es inferior a las de los diseños óptimos tradicionales.

Pese a que han sido incluidas en la bibliografía que aparece al final de estas páginas , por completitud de este capítulo ha de decirse aquí que estos resultados han sido publicados en:

1. Mayhua–López, E., Gómez–Verdejo, V., and Figueiras–Vidal, A. R. (2010). Improving boosting performance with a local combination of learners. In *Proc. Intl. Joint Conf. on Neural Networks*, pp. 1983–1990, Barcelona.

   Se trata de una versión preliminar, que ya permitía entrever que la idea de incluir puertas en la fusión de aprendices era prometedora.

2. Mayhua–López, E., Gómez–Verdejo, V., and Figueiras–Vidal, A. R. (2012). Real adaboost with gate controlled fusion. *IEEE Trans. Neural Networks Learning Systems*, 23(12):2003–2009.

   Este artículo presenta ya los resultados finales, pudiendo decirse que es una versión concisa del capítulo 3.

En segundo lugar, también se ha constatado que es posible obtener diseños de prestaciones ventajosas a partir de aprendices tipo Máquinas de Vectores Soporte, cuya estabilidad dificulta grandemente su incorporación a conjuntos tipo Boosting; habiendo tenido éxito limitado los intentos previos de atenuar esa estabilidad —o, en general, de combatir la limitación— que han visto la luz en la literatura especializada. La forma en que se ha conseguido reducir esta dificultad consiste en combinar un algoritmo de diseño, el de Programación Lineal, que permite un mejor control del entrenamiento de los aprendices, con una cuidadosamente elegida forma de submuestrear las instancias de entrenamiento para elegir las que pueden pasar a formar parte de cada unidad. Las prestaciones obtenidas mediante esta vía compuesta, detalladas en el capítulo 4, son, en términos generales, mejores que las proporcionadas por máquinas individuales y diseños previos basados en Vectores Soporte.

Una versión resumida de lo expuesto en el capítulo 4 constituye el artículo

3. Mayhua–López, E., Gómez–Verdejo, V., and Figueiras–Vidal, A. R. (2013). Boosting ensembles with subsampled lpsvm learners. Remitido a *IEEE Trans. Neural Networks Learning Systems*;

que se encuentra en la fase final de una segunda revisión, con altos visos de ser aceptado para publicación en fechas próximas.

5.2. Lineas de investigación abiertas

De todo trabajo de investigación que culmine con resultados favorables emergen posibles líneas de trabajo a seguir, que tienen diferentes orientaciones. Las más inmediatas, generalizaciones o extensiones que simplemente requieren aplicar a lo descubierto principios y técnicas previamente disponibles, con el objeto de cubrir problemas o aplicaciones que no han sido directamente abordadas en la tarea que se ha llevado a cabo. Otro nivel corresponde a las direcciones en que lo descubierto puede expandirse para obtener ventajas adicionales a las por ello aportadas. De acuerdo con lo anterior presentaremos las líneas de investigación abiertas por las aportaciones contenidas en esta Tesis Doctoral.

5.2.1. Generalizaciones y extensiones inmediatas

a) En este trabajo, se ha aplicado un solo algoritmo básico para la selección de los centroides de las puertas. Como queda de manifiesto por la bibliografía citada en los lugares correspondientes, hay otros muchos procedimientos establecidos con objeto análogo. Considerar la eficacia de las más representativos y determinar en qué clase de situaciones pueden implicar ventaja es una línea de trabajo inmediata, pero no por ello de menor valor práctico.

b) También resulta inmediato indagar sobre la posibilidad de abordar la extensión de lo aquí analizado a esquemas Boosting distintos al Real AdaBoost. A guisa de ejemplo, no sería irrelevante comprobar qué resulta de combinar las contribuciones de esta Tesis con aproximaciones Boosting que pretenden combatir otras dificultades intrínsecas a las versiones básicas de estos algoritmos, como la sobreactuación del énfasis convencional en muchos casos prácticos. Así, extender al modo del Real AdaBoost los algoritmos de énfasis híbrido (según proximidad a la frontera y nivel de error) propuestos en [Gómez-Verdejo et al., 2006, Gómez-Verdejo et al., 2008] podría no solo pro-

porcionar diseños de alto interés práctico, sino ilustrar sobre las relaciones existentes entre la sobreenfatización y las limitaciones estructurales.
Como es fácil de comprender, lo importante de estos posibles estudios es verificar que no se produce una reducción de las ventajas de cada una de las técnicas por el hecho de agruparlas.

c) Otras extensiones de consideración inmediata serian las consistentes en emplear aprendices tipo Máquinas de Vectores Soporte de última generación, como los de múltiples núcleos [Gönen and Alpaydın, 2011], a los diseños llevados a cabo en el capítulo 4. Dado que los núcleos múltiples combaten las limitaciones del núcleo único (p. ej., el carácter estacionario paso-bajo que implica el uso de los tradicionales núcleos gaussianos), el éxito de esta línea supondría la posibilidad de disponer de componentes de altas prestaciones para problemas no paso-bajo o/y no estacionarios.
También, puesto que se trata de diseños que siguen el principio de Máximo Margen, podría pensarse en extender las formulaciones y el análisis a diseños de conjuntos cuyos aprendices fuesen versiones debilitadas de las recientemente introducidas combinaciones con Pesos Funcionales Generados por Puerta [Omari and Figueiras-Vidal, 2013], cuyas favorables características se evidencian en el artículo citado.

d) No ha de omitirse la mención de la posibilidad de aplicar los procedimientos de reducción de dimensiones y diseño de modelos dispersos en este ámbito. Las técnicas de ambos tipos [Guyon and Elisseeff, 2003, Guyon, 2006] vienen proporcionando éxito en muchos planteamientos y aplicaciones —desde mejores prestaciones hasta sensible reducción de la carga operativa— como para no considerarlas en este contexto.

e) No debe revestir dificultad teórica importante la extensión de la formulación aquí introducida a problemas de tipo más general, como son:

- la clasificación M-aria;
- los problemas desequilibrados o sensibles al coste;
- los problemas multietiqueta;

etc. Se renuncia a una enumeración completa, ante la gran variedad de las opciones posibles.

5.2.2. Ampliaciones de mayor perspectiva

a) Cabe la posibilidad de ampliar los conceptos implícitos en los esquemas aquí propuestos a otros tipos de conjuntos de máquinas:

   - métodos de Bagging/Wagging: inclusión de puertas o/y incorporación de unidades tipo Vectores Soporte;
   - agregaciones múltiples basadas en diversidad: considerando esquemas basados en núcleos e incluyendo puertas.

b) Así como la exportación de estas metodologías a la resolución de otras categorías de problemas que, sin embargo, comparten elementos comunes con los aquí estudiados, como son:

   - problemas de conjuntos de máquinas para regresión [Freund and Schapire, 1995, Bailly and Milgram, 2009];
   - problemas de detección de novedad [Rätsch et al., 2002a];
   - problemas multi-tarea [Caruana, 1997, Chapelle et al., 2011].

5.3. Conclusiones

Los trabajos aquí expuestos acreditan que es posible mejorar diseños (en particular, de conjuntos de máquinas) de muy altas prestaciones reexaminando sus funda-

mentos y reelaborando la forma de realizarlos mediante la inclusión de modificaciones "ad hoc" para hacerlos más resistentes a las limitaciones que muestren. Sin que ello signifique cambio de paradigma (cooperación–competición), ni siquiera adopción de principios diferentes (combinación, énfasis), este proceder puede dar lugar a mejoras sensibles en diseños de conjuntos de máquinas, tanto en términos generales cuanto en aplicaciones particulares, como se ha visto en la utilización de puertas en la agregación Boosting o en el recurso a la Programación Lineal y el submuestreo para agregar aprendices estables (Máquinas de Vectores Soporte) en conjuntos Boosting.

Naturalmente, el camino seguido —analizar, reflexionar, reformular y ajustar— tendrá segura utilidad en otros muchos ámbitos del actual estado del arte del diseño de conjuntos de máquinas de aprendizaje, y es perfectamente combinable con otros procedimientos más radicales para conseguir diseños innovadores.

Apéndice A

Fórmulas del algoritmo RAB

A.1. Selección de los pesos de salida

Para obtener los valores de los pesos de salida asociados a cada uno de los clasificadores base, en cada ronda se seleccionará el valor de α_t que minimice la cota del error de entrenamiento dada en (2.9).

Para simplificar el desarrollo matemático, se denotará el producto $f_t(\mathbf{x}^{(l)})y^{(l)}$ mediante $u_t(\mathbf{x}^{(l)})$, de modo que la cota a minimizar queda expresada como

$$B_t \leq \frac{1}{L}\sum_{l=1}^{L}\exp[-F_{t-1}(\mathbf{x}^{(l)})y^{(l)}]\left\{\frac{1+u_t(\mathbf{x}^{(l)})}{2}\exp(-\alpha_t)+\frac{1-u_t(\mathbf{x}^{(l)})}{2}\exp(\alpha_t)\right\}. \quad \text{(A.1)}$$

Para calcular el valor de α_t que minimiza el segundo término de (A.1), se deriva dicha expresión respecto de α_t y se iguala a 0. El resultado es:

$$\frac{1}{L}\sum_{l=1}^{L}\exp[-F_{t-1}(\mathbf{x}^{(l)})y^{(l)}]\left\{-\frac{1+u_t(\mathbf{x}^{(l)})}{2}\exp(-\alpha_t)+\frac{1-u_t(\mathbf{x}^{(l)})}{2}\exp(\alpha_t)\right\}=0 \quad \text{(A.2)}$$

Despejando el término exponencial dependiente de α_t se obtiene

$$\exp(2\alpha_t)=\frac{\sum_{l=1}^{L}[1+u_t(\mathbf{x}^{(l)})]\exp[-F_{t-1}(\mathbf{x}^{(l)})y^{(l)}]}{\sum_{l=1}^{L}[1-u_t(\mathbf{x}^{(l)})]\exp[-F_{t-1}(\mathbf{x}^{(l)})y^{(l)}]} \quad \text{(A.3)}$$

Sí, además, se combina la expresión de B_t en (2.6) para el instante $t-1$,

$$B_{t-1} = \frac{1}{L}\sum_{l=1}^{L} \exp[-F_{t-1}(\mathbf{x}^{(l)})y^{(l)}] \tag{A.4}$$

con la función de énfasis, $D_t(\mathbf{x}^{(l)})$, en función de la salida parcial del conjunto correspondiente a la ronda t-ésima, se obtiene

$$D_t(\mathbf{x}^{(l)}) = \frac{\exp(-F_{t-1}(\mathbf{x}^{(l)})y^{(l)})}{\sum_{l=1}^{L}\exp(-F_{t-1}(\mathbf{x}^{(l)})y^{(l)})} = \frac{1}{LB_{t-1}}\exp[-F_{t-1}(\mathbf{x}^{(l)})y^{(l)}] \tag{A.5}$$

De las ecuaciones (A.3) y (A.5) se obtiene la siguiente expresión

$$\exp(2\alpha_t) = \frac{\sum_{l=1}^{L}[1+u_t(\mathbf{x}^{(l)})]LB_{t-1}D_t(\mathbf{x}^{(l)})}{\sum_{l=1}^{L}[1-u_t(\mathbf{x}^{(l)})]LB_{t-1}D_t(\mathbf{x}^{(l)})} \tag{A.6}$$

que se reescribe para llegar a

$$\exp(2\alpha_t) = \frac{LB_{t-1}+LB_{t-1}\sum_{l=1}^{L}u_t(\mathbf{x}^{(l)})D_t(\mathbf{x}^{(l)})}{LB_{t-1}-LB_{t-1}\sum_{l=1}^{L}u_t(\mathbf{x}^{(l)})D_t(\mathbf{x}^{(l)})} = \frac{1+\gamma_t}{1-\gamma_t}. \tag{A.7}$$

Finalmente, tomando logaritmos en (A.7) se obtiene la expresión para el cálculo de α_t:

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1+\gamma_t}{1-\gamma_t}\right). \tag{A.8}$$

A.2. Convergencia del error de entrenamiento

La demostración de este resultado para el algoritmo RAB se realiza partiendo de la cota sobre E_t dada por (2.6) y (A.1). Operando los términos entre llaves, se obtiene

$$B_t \leq \frac{1}{2L}\sum_{l=1}^{L}\exp[-F_{t-1}(\mathbf{x}^{(l)})y^{(l)}]\left\{[\exp(-\alpha_t)+\exp(\alpha_t)]+u_t(\mathbf{x}^{(l)})\left[\exp(-\alpha_t)-\exp(\alpha_t)\right]\right\} \tag{A.9}$$

Haciendo uso de la expresión para α_t dada en (A.8), se agrupan los términos exponenciales, obteniendo así

$$\exp(-\alpha_t)+\exp(\alpha_t) = \sqrt{\frac{1-\gamma_t}{1+\gamma_t}}+\sqrt{\frac{1+\gamma_t}{1-\gamma_t}} = \frac{1-\gamma_t+1+\gamma_t}{\sqrt{1-\gamma_t^2}} = \frac{2}{\sqrt{1-\gamma_t^2}} \tag{A.10}$$

$$\exp(-\alpha_t) - \exp(\alpha_t) = \sqrt{\frac{1-\gamma_t}{1+\gamma_t}} - \sqrt{\frac{1+\gamma_t}{1-\gamma_t}} = \frac{1-\gamma_t-(1+\gamma_t)}{\sqrt{1-\gamma_t^2}} = \frac{-2\gamma_t}{\sqrt{1-\gamma_t^2}} \quad \text{(A.11)}$$

Sustituyendo estos resultados en (A.9) y operando, se llega a

$$\begin{aligned} B_t &\leq \frac{1}{2L}\sum_{l=1}^{L}\exp[-F_{t-1}(\mathbf{x}^{(l)})y^{(l)}]\left\{\frac{2}{\sqrt{1-\gamma_t^2}} + u_t(\mathbf{x}^{(l)})\frac{-2\gamma_t}{\sqrt{1-\gamma_t^2}}\right\} = \\ &= \frac{1}{\sqrt{1-\gamma_t^2}}\frac{1}{L} \quad \sum_{l=1}^{L}\exp[-F_{t-1}(\mathbf{x}^{(l)})y^{(l)}]\left\{1-u_t(\mathbf{x}^{(l)})\gamma_t\right\} = \\ &= \frac{1}{\sqrt{1-\gamma_t^2}}\left\{\frac{1}{L}\sum_{l=1}^{L}\exp[-F_{t-1}(\mathbf{x}^{(l)})y^{(l)}] - \gamma_t\frac{1}{L}\sum_{l=1}^{L}\exp[-F_{t-1}(\mathbf{x}^{(l)})y^{(l)}]u_t(\mathbf{x}^{(l)})\right\} \quad \text{(A.12)} \end{aligned}$$

Esta cota puede reescribirse de manera más compacta sustituyendo las expresiones de B_{t-1} y γ_t, dadas por (A.4) y (A.5) respectivamente, en la ecuación anterior

$$B_t \leq \frac{1}{\sqrt{1-\gamma_t^2}} \quad \left\{B_{t-1} - B_{t-1}\gamma_t^2\right\} = \sqrt{1-\gamma_t^2} \quad B_{t-1} \quad \text{(A.13)}$$

y, aplicando recursivamente la desigualdad, se llega a

$$B_t \leq \prod_{t'=1}^{t}\sqrt{1-\gamma_{t'}^2}. \quad \text{(A.14)}$$

De ello se tiene la siguiente cota del error de entrenamiento en la iteración t-ésima:

$$E_t = \frac{1}{2L}\sum_{l=1}^{L} \mid \text{sign}\left[F_t(\mathbf{x}^{(l)})\right] - y^{(l)} \mid \leq \prod_{t'=1}^{t}\sqrt{1-\gamma_{t'}^2}. \quad \text{(A.15)}$$

Apéndice B

Procedimiento para determinar el valor de K

Shin y Cho en [Shin and Cho, 2003b] proponen un procedimiento que permite fijar el valor de los K vecinos más próximos al momento de seleccionar las muestras que se localizan cerca de la frontera de decisión. Para ello, se define como región de superposición, $\mathbf{R}$, al espacio de entrada donde se localizan muestras de entrenamiento de ambas clases. Sí $\mathbf{D}$ es el conjunto de las L muestras de entrenamiento, la intersección de $\mathbf{D}$ con $\mathbf{R}$ genera el subconjunto de v muestras de entrenamiento que pertenecen a la región de superposición.

Según los mismos autores, la probabilidad con que las v muestras caigan en la región $\mathbf{R}$ sigue una distribución binomial

$$P_r(v) = \frac{L!}{v!(L-v)!} \left(P_R(\mathbf{x})\right)^v \left(1 - P_R(\mathbf{x})\right)^{L-v}, \tag{B.1}$$

donde $P_R(\mathbf{x})$ es la probabilidad con que una muestra $\mathbf{x}$ pertenece a la región $\mathbf{R}$, por lo tanto, v se puede calcular como:

$$v = LP_R(\mathbf{x}). \tag{B.2}$$

El valor de $P_R(\mathbf{x})$ para la muestra $\mathbf{x}$ viene dada por

$$P_R(\mathbf{x}) = \sum_{j=1}^{2} P(\mathbf{x} \in \mathbf{R}, \omega_j) \tag{B.3}$$

osea, $P(\mathbf{x} \in \mathbf{R}, \omega_j)$ es la probabilidad conjunta de pertenencia de la muestra $\mathbf{x}$ a la clase ω_j y la región $\mathbf{R}$.

Si la región $\mathbf{R}$ se divide en $\mathbf{R1}$ para un lado de la frontera de decisión y $\mathbf{R2}$ para el lado contrario, (B.3) se puede reescribir como:

$$\begin{aligned} P_R(\mathbf{x}) &= P(\mathbf{x} \in \mathbf{R}, \omega_1) + P(\mathbf{x} \in \mathbf{R}, \omega_2) \\ &= P(\mathbf{x} \in \mathbf{R1} \cup \mathbf{R2}, \omega_1) + P(\mathbf{x} \in \mathbf{R1} \cup \mathbf{R2}, \omega_2) \\ &= (P(\mathbf{x} \in \mathbf{R1}, \omega_2) + P(\mathbf{x} \in \mathbf{R2}, \omega_1)) + (P(\mathbf{x} \in \mathbf{R1}, \omega_1) + P(\mathbf{x} \in \mathbf{R2}, \omega_2)) \,. \end{aligned} \tag{B.4}$$

El primer y segundo paréntesis, indican la probabilidad con que las muestras que se localizadas en la región $\mathbf{R}$ están incorrecta y correctamente clasificadas, respectivamente. Asumiendo que, tanto $\mathbf{R1}$ como $\mathbf{R2}$ tienen, aproximadamente, el mismo número de muestras correcta e incorrectamente clasificadas, los valores de las probabilidades de los dos paréntesis es la misma. Además, la región $\mathbf{R}$ contienen a todas las muestras superpuestas, eso quiere decir que el primer paréntesis es el error de clasificación, P_e, sobre todo el conjunto de entrenamiento. Por lo tanto (B.4) se puede simplificar a

$$P_R(\mathbf{x}) = 2P_e \tag{B.5}$$

y agrupando (B.2) con (B.5) se tiene que

$$v = 2LP_e. \tag{B.6}$$

Finalmente, se elige K como:

$$K = \text{mín}\left\{K' | b_{K'} \geq v, K' = 2, \ldots, L-1\right\}, \tag{B.7}$$

donde $b_{K'}$ es el número de muestras que cumplen la condición

$$pr(\mathbf{x}^{(l)}) > 0 \quad \text{y} \quad co(\mathbf{x}^{(l)}) \geq \frac{1}{2} \tag{B.8}$$

para $K' = 2, \ldots, L-1$.

Tanto $pr(\mathbf{x}^{(l)})$ como $co(\mathbf{x}^{(l)})$ se calculan del mismo modo que en el Apartado 3.2.1, es decir

$$pr(\mathbf{x}^{(l)}) = \sum_j P_j(\mathbf{x}^{(l)}) \log_2 \frac{1}{P_j(\mathbf{x}^{(l)})}. \tag{B.9}$$

y

$$co(\mathbf{x}^{(l)}) = P_{y^{(l)}}(\mathbf{x}^{(l)}). \tag{B.10}$$

Apéndice C

Descripción de las bases de datos

En este apéndice comentamos las características de las bases de datos que se han utilizado para evaluar las prestaciones de los diferentes algoritmos de clasificación propuestos en esta Tesis. Los conjuntos de datos han sido seleccionados de modo que provean suficiente diversidad en cuanto a número de muestras, atributos y complejidad, razón por la cual se consideran tanto problemas reales como artificiales.

Se han elegido once bases de datos pertenecientes al repositorio de la Universidad de California en Irvine (UCI) [Frank and Asuncion, 2010]; concretamente los correspondientes a los problemas:

- *Abalone*: originalmente este problema consiste en estimar la edad de un tipo de caracol marino ("abalone") a partir de una serie de medidas físicas sobre su concha (longitud, diámetro, altura, peso, etc.); sin embargo, se ha transformado en un problema de clasificación binaria, tal y como se indica en [Ruiz and de Teruel, 2001], para determinar si la edad del molusco es superior o inferior a los 10 años.

- *Breast*: "Wisconsin Breast Cancer" es un problema que consiste en determinar el carácter benigno o maligno de un tumor a partir de una serie de atributos

medidos en los núcleos de las células cancerígenas. Las tasas de error mínimas registradas empíricamente están alrededor de 2.1 %, según se registra en [Frank and Asuncion, 2010].

- *Contraceptive*: este problema tiene como objetivo predecir, a partir de las características demográficas y socioeconómicas de un grupo de mujeres casadas, si emplean o no algún método anticonceptivo.

- *Diabetes*: La base de datos "Pima Indian Diabetes", cuenta con información de mujeres indígenas proveniente de la tribu Pima de Arizona, de 21 años o más de edad. A los indios Pima se les conoce por estar genéticamente predispuestos a padecer diabetes. La base de datos está compuesta por 768 instancias, con un conjunto de 8 variables o atributos: número de embarazos, concentración plasmática de glucosa a las 2 horas de una prueba de tolerancia a la glucosa oral, presión diastólica de la sangre, espesor de la piel del tríceps, cantidad de insulina en dos horas, índice de masa corporal, antecedentes familiares de padecer diabetes y, la edad del paciente. La etiqueta es el diagnóstico de la condición del individuo.

- *Hepatitis*: el propósito de esta base de datos es predecir la presencia o ausencia de hepatitis en pacientes que han sido sometidos a diversos análisis clínicos. Esta base de datos contiene 19 atributos, que han sido extraídos de un conjunto de 155 pacientes, de los cuales 32 son pacientes que han fallecido.

- *Duke*: este conjunto de datos fue construido por la Duke University y Duke Medical Center. El correspondiente problema consiste en predecir el estado clínico del cáncer de mama humano mediante el uso de la expresión de genes pro-les. La base de datos originalmente se compone de 49 muestras separadas en dos clases: 25 muestras que dieron positivo a la recepción de estrógeno y 24 resultaron negativos. Para cada una de estas muestras se midieron los niveles de expresión de 7128 genes.

- *Ionosfera*: el objetivo es determinar si la señal radar que se refleja en la ionosfera indica o no la presencia allí de algún tipo de estructura. Las tasas de error mínimas registradas empíricamente están alrededor de 1.8 %, según se indica en [Frank and Asuncion, 2010].

- *Madelon*: es una base de datos generada de forma sintética. Las muestras se agrupados en 32 grupos situados en los vértices de un hipercubo de cinco dimensiones etiquetados aleatoriamente con +1 ó -1. Las cinco dimensiones constituyen cinco características informativas. Se añaden 15 combinaciones lineales de dichas características para formar un conjunto de 20 rasgos. Se añaden una serie de características denominados "sondas" que no tienen poder predictivo. Finalmente, las características y las muestras se ordenan de manera aleatoria. Esta base de datos ha sido utilizada en el reto de selección de características del NIPS 2003 (NIPS, "Neural Information Processing Systems") [Guyon et al., 2004].

- *Thyroid*: en este caso, la tarea a resolver consiste en determinar si un paciente tiene el funcionamiento de la tiroides en condiciones normales o no. Éste es un problema con una mayor cantidad de muestras correspondientes a pacientes que poseen la tiroides funcionando en condiciones por debajo de lo normal, es decir, un problema con poblaciones desequilibradas.

- *Twonorm*: este es un problema sintético propuesto por Leo Breiman [Breiman, 1996b]. Los datos de cada clase proceden de una distribución normal multivariable con varianza unidad. La dimensión de los datos es 20 y una de las clases tiene como media el vector $[a \quad a \quad \ldots \quad a]$ y la clase contraria $[-a \quad -a \quad \ldots \quad -a]$, con $a = 2/\sqrt{20}$.

- *Waveform*: contiene datos de tres clases, donde cada clase procede de la combinación lineal aleatoria de dos de tres ondas "base" triangulares, a las que se les ha añadido ruido Gaussiano de media cero y varianza unidad

[Breiman et al., 1984]. Para transformar el problema de tres clases en binario, se agrupan dos clases en una con el objeto de distinguir si el dato procede o no de la combinación de dos de las ondas "base".

Las bases de datos restantes proceden de otras fuentes. Son:

- *Crabs*: describe cinco mediciones morfológicas de cincuenta cangrejos de ambos sexos y dos colores, pertenecientes a la variedad Leptograpsus Variegatus. El objetivo es clasificar los cangrejos según su sexo. Aparece en [Ripley, 1996].

- *Kwok*: esta es una base de datos sintética creada por [Kwok, 1999]. Los datos corresponden a cinco gaussianas bidimensionales esfericas: dos gaussianas para generar los datos de una clase y tres para la clase contraria. La tasa de error de Bayes sobre el conjunto de prueba es del 11.3 %:

- *Ripley*: es una base de datos binaria propuesta en [Ripley, 1994]. Los datos de cada clase proceden de distribuciones formadas por las mezclas de dos gaussianas de igual matriz de covarianza y probabilidad a priori. La tasa de error de Bayes sobre su conjunto de prueba es del 8 %.

Apéndice D

Test de diferencias estadísticas

Para comparar los errores de clasificación entre dos algoritmos resulta interesante saber en qué casos sus poblaciones de muestras son estadísticamente diferentes; para ello, hemos aplicado el T-test y calculado el p-valor como indicador de diferencia estadística entre los resultados [Hill and Lewicki, 2006].

El uso de los test de diferencias estadísticas requiere que las realizaciones sean independientes. Aunque en los casos en los que se va aplicar se emplean los mismos datos de entrenamiento, y aunque se inicializan de forma distinta los pesos de los MLPs e, incluso, las particiones de los datos de entrenamiento es aleatorio, no se puede garantizar esta independencia. Por tanto, los test que se obtienen son solo un indicativo.

D.1. T-test

Este test proporciona un valor, t, que resulta de calcular la diferencia entre las medias de las dos distribuciones que se comparan normalizada por sus desviaciones estándares; dado que se desconoce el tipo de distribución que siguen los grupos

de datos a comparar, el T-test considera que sus medias siguen una distribución gaussiana de media $\{m_i\}_{i=1,2}$ (dada por la media muestral) y desviación estándar $\{\sigma_i/\sqrt{N_i}\}_{i=1,2}$, donde las desviaciones estándares $\{\sigma_i\}_{i=1,2}$ siguen una distribución Chi-cuadrado y los valores $\{N_i\}_{i=1,2}$ representan al número de valores en cada grupo de datos. De este modo, el valor de t viene dado por

$$t = \frac{m_1 - m_2}{\sqrt{\sigma_1^2/N_1 + \sigma_2^2/N_2}} \tag{D.1}$$

y, tal y como se puede demostrar, se distribuye según una t de Student con N_1+N_2-1 grados de libertad.

En el caso de los experimentos que se presentan en esta Tesis Doctoral, los valores m_1 y m_2 se corresponden con los valores de CE de los algoritmos que se comparan y, dado que cada uno de estos valores se ha promediado sobre 50 iteraciones, el número de grados de libertad de la distribución t de Student será 99.

Entonces, y de acuerdo con las tablas de la distribución t de Student, se puede afirmar con un 95 % de seguridad que los valores medios de las dos distribuciones son estadísticamente diferentes si $|t| > 1{,}66$.

Para extraer conclusiones con un nivel de certeza distinto del 95 %, podrían emplearse los valores límite correspondientes, algunos de los cuales se encuentran recogidos en el Cuadro D.1.

Tabla D.1: Valores límite del parámetro t del T-test para distintos niveles de certeza.

Certeza (%)	75	80	85	90	**95**	97.5	99	99.5	99.9	99.95
t_{limit}	0.677	0.845	1.042	1.290	**1.660**	1.984	2.364	2.626	3.174	3.390

D.2. El p-valor

El p-valor o nivel de significación observado, es el área de la cola de una distribución normal estándar (o varias de colas si el test es bilateral) definida a partir del

estadístico de contraste. En el caso de esta Tesis Doctoral el estadístico de contraste es el resultado de aplicar el T-test. Este valor se interpreta como una medida de la evidencia estadística que las poblaciones aportan a favor de la hipótesis alternativa (existe diferencia significativa) o en contra de la hipótesis nula (no existe diferencia significativa). Cuando el p-valor es pequeño, se considera que hay una fuerte evidencia a favor de la hipótesis alternativa. En la práctica se suele adoptar el criterio de aceptar la hipótesis nula cuando el p-valor es mayor que el nivel de significación, que se prefija normalmente a 0.05 para un nivel de certeza del 95 %.

Bibliografía

[Almeida et al., 2000] Almeida, M. B., Braga, A. P., and Braga, J. P. (2000). SVM-KM: Speeding SVMs learning with a priori cluster selection and k-means. In *Proc. 6th Brazilian Symp. Neural Networks*, pp. 162–167, Rio de Janeiro.

[Bailly and Milgram, 2009] Bailly, K. and Milgram, M. (2009). 2009 special issue: Boosting feature selection for neural network based regression. *Neural Networks*, 22(5-6):748–756.

[Barber, 2012] Barber, D. (2012). *Bayesian Reasoning and Machine Learning*. Cambridge University Press, Cambridge, UK.

[Barkat, 2005] Barkat, M. (2005). *Signal detection and estimation*. Artech House, Boston, MA.

[Bishop, 1995] Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, New York, NY.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag, New York, NY.

[Bradley and Schapire, 2007] Bradley, J. K. and Schapire, R. E. (2007). Filterboost: Regression and classification on large datasets. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T., editors, *Advances in Neural Information Processing Systems 20*, Cambridge, MA. MIT Press.

[Bradley and Mangasarian, 1998] Bradley, P. S. and Mangasarian, O. L. (1998). Feature selection via concave minimization and support vector machines. In *Proc. 15th Intl. Conf. on Machine Learning*, pp. 82–90, San Francisco, CA. Morgan Kaufmann.

[Breiman, 1996a] Breiman, L. (1996a). Bagging predictors. *Machine Learning*, 24(2):123–140.

[Breiman, 1996b] Breiman, L. (1996b). Bias, variance and arcing classifiers. Technical Report 460, University of California, Statistics Department, Berkeley, CA.

[Breiman, 1998] Breiman, L. (1998). Arcing classifiers. *The Annals of Statistics*, 26(3):801–824.

[Breiman, 1999] Breiman, L. (1999). Prediction games and arcing algorithms. *Neural Computation*, 11:1493–1517.

[Breiman, 2000] Breiman, L. (2000). Randomizing outputs to increase prediction accuracy. *Machine Learning*, 40:229–242.

[Breiman et al., 1984] Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA.

[Buntine, 1996] Buntine, W. (1996). A guide to the literature on learning probabilistic networks from data. *IEEE Trans. Knowledge and Data Engineering*, 8(2):195–210.

[Caruana, 1997] Caruana, R. (1997). Multitask learning. *Machine Learning*, 28(1):41–75.

[Chang and Lippmann, 1993] Chang, E. I. and Lippmann, R. P. (1993). A boundary hunting radial basis function classifier which allocates centers constructively. In Hanson, S. J., Cowan, J. D., and Giles, C. L., editors, *Advances in Neural Information Processing Systems 5*, pp. 139–146, San Mateo, CA. Morgan Kaufmann.

[Chapelle et al., 2011] Chapelle, O., Shivaswamy, P., Vadrevu, S., Weinberger, K., Zhang, Y., and Tseng, B. (2011). Boosted multi-task learning. *Machine Learning*, 85(1-2):149–173.

[Chen and Wang, 2011] Chen, K. and Wang, S. (2011). Semi-supervised learning via regularized boosting working on multiple semi-supervised assumptions. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 33(1):129–143.

[Chen, 1995] Chen, S. (1995). Nonlinear time series modelling and prediction using Gaussian RBF networks with enhanced clustering and RLS learning. *Electronics Letters*, 31, no. 2(2):117–118.

[Chen et al., 1991] Chen, S., Cowan, C. F. N., and Grant, P. M. (1991). Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Trans. Neural Networks*, 2(2):302–309.

[Chen et al., 2010] Chen, S., He, H., and Garcia, E. (2010). RAMOBoost: Ranked Minority Oversampling in Boosting. *IEEE Trans. Neural Networks*, 21(10):1624 –1642.

[Chen et al., 2006] Chen, S., Wang, X., Hong, X., and Harris, C. (2006). Kernel classifier construction using orthogonal forward selection and boosting with Fisher ratio class separability measure. *IEEE Trans. Neural Networks*, 17(6):1652–1656.

[Collins et al., 2002] Collins, M., Schapire, R. E., and Singer, Y. (2002). Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, 48(1-3):253–285.

[Cybenko, 1989] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society B*, 39(1):1–38.

[Dietterich, 2000] Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40:139–157.

[Drucker and Cortes, 1996] Drucker, H. and Cortes, C. (1996). Boosting decision trees. In Touretzky, D. S., Mozer, M., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*, pp. 479–485, Cambridge, MA. MIT Press.

[Drucker et al., 1993] Drucker, H., Schapire, R. E., and Simard, P. (1993). Boosting performance in neural networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 7:705–719.

[Duda et al., 2001] Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern classification*. Wiley, New York, NY.

[Duffy and Helmbold, 1999] Duffy, N. and Helmbold, D. P. (1999). Potential boosters? In Solla, S. A., Leen, T. K., and Müller, K.-R., editors, *Advances in Neural Information Processing Systems, 12*, pp. 258–264, Cambridge, MA. MIT Press.

[Eads et al., 2009] Eads, D., Rosten, E., and Helmbold, D. P. (2009). Learning object location predictors with boosting and grammar-guided feature extraction. In *Proc. British Machine Vision Conference*, London, UK. British Machine Vision Association.

[Elanayar and Shin, 1994] Elanayar, S. and Shin, Y. C. (1994). Radial basis function neural network for approximation and estimation of nonlinear stochastic dynamic systems. *IEEE Trans. Neural Networks*, 5(4):594 –603.

[Frank and Asuncion, 2010] Frank, A. and Asuncion, A. (2010). UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences.

[Freund, 1995] Freund, Y. (1995). Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285.

[Freund, 2001] Freund, Y. (2001). An adaptive version of the boost by majority algorithm. *Machine Learning*, 43:293–318.

[Freund and Schapire, 1995] Freund, Y. and Schapire, R. E. (1995). A decision-theoretic generalization of on-line learning and an application to boosting. In *Proc. of the Second European Conf. on Computational Learning Theory*, pp. 23–37, London, UK. Springer-Verlag.

[Freund and Schapire, 1996] Freund, Y. and Schapire, R. E. (1996). Experiments with a New Boosting Algorithm. In *In Proc. of the 13th International Conference on Machine Learning*, pp. 148–156, Bari, Italy.

[Freund and Schapire, 1997] Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139.

[Friedman, 1997] Friedman, J. H. (1997). On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Min. Knowl. Discov.*, 1(1):55–77.

[Friedman, 2002] Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367 – 378.

[Friedman et al., 2000] Friedman, J. H., Hastie, T., and Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 28(2):337–407.

[Fritzke, 1994] Fritzke, B. (1994). Supervised learning with growing cell structures. In Cowan, J. D., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6*, pp. 255–262, San Mateo, CA. Morgan Kaufmann.

[García-Pedrajas et al., 2007] García-Pedrajas, N., García-Osorio, C., and Fyfe, C. (2007). Nonlinear boosting projections for ensemble construction. *Journal of Machine Learning Research*, 8:1–33.

[Ge and Jiang, 2006a] Ge, Y. and Jiang, W. (2006a). A note on mixtures of experts for multiclass responses: approximation rate and consistent Bayesian inference. In *Proc. of the 23rd International Conference on Machine Learning*, pp. 329–335, New York, NY. ACM Pres.

[Ge and Jiang, 2006b] Ge, Y. and Jiang, W. (2006b). On consistency of Bayesian inference with mixtures of logistic regression. *Neural Computation*, 18:224–243.

[Gómez-Verdejo et al., 2008] Gómez-Verdejo, V., Arenas-García, J., and Figueiras-Vidal, A. R. (2008). A dynamically adjusted mixed emphasis method for building boosting ensembles. *IEEE Trans. Neural Networks*, 19(1):3 –17.

[Gómez-Verdejo et al., 2006] Gómez-Verdejo, V., Ortega-Moral, M., Arenas-García, J., and Figueiras-Vidal, A. R. (2006). Boosting by weighting critical and erroneous samples. *Neurocomputing*, 69(7-9):679 – 685.

[Gönen and Alpaydın, 2011] Gönen, M. and Alpaydın, E. (2011). Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12:2211–2268.

[Grove and Schuurmans, 1998] Grove, A. J. and Schuurmans, D. (1998). Boosting in the limit: maximizing the margin of learned ensembles. In *Proc. of the 15th National/Tenth conf. on Artificial intelligence/Innovative applications of artificial intelligence*, AAAI '98/IAAI '98, pp. 692–699, Menlo Park, CA. American Association for Artificial Intelligence.

[Guyon, 2006] Guyon, I. (2006). *Feature Extraction: Foundations and Applications*. Springer, London, UK.

[Guyon and Elisseeff, 2003] Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182.

[Guyon et al., 2004] Guyon, I., Gunn, S. R., Ben-Hur, A., and Dror, G. (2004). Result analysis of the NIPS 2003 feature selection challenge. In *Advances in Neural Information Processing Systems 16*.

[Hansen and Salamon, 1990] Hansen, L. K. and Salamon, P. (1990). Neural network ensembles. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(10):993–1001.

[Haykin, 2007] Haykin, S. (2007). *Neural Networks: A Comprehensive Foundation (3rd Edition)*. Prentice-Hall, Upper Saddle River, NJ.

[Hill and Lewicki, 2006] Hill, T. and Lewicki, P. (2006). *Statistics: Methods and Applications : a Comprehensive Reference for Science, Industry, and Data Mining*. StatSoft.

[Hornik, 1991] Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257.

[Hwang and Bang, 1996] Hwang, Y.-S. and Bang, S.-Y. (1996). An efficient method to construct a radial basis function neural network classifier and its application to unconstrained handwritten digit recognition. In *Proc. Intl. Conf. on Pattern Recognition*, pp. 640–644, Vienna, Austria.

[Jacobs et al., 1991] Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3:79–87.

[Jacobs et al., 1997] Jacobs, R. A., Peng, F., and Tanner, M. A. (1997). A Bayesian approach to model selection in hierarchical mixtures-of-experts architectures. *Neural Networks*, 10:231–241.

[Jordan and Jacobs, 1994] Jordan, M. I. and Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214.

[Jordan and Xu, 1995] Jordan, M. I. and Xu, L. (1995). Convergence results for the EM approach to mixtures of experts architectures. *Neural Networks*, 8:1409–1431.

[Kalal et al., 2008] Kalal, Z., Matas, J., and Mikolajczyk, K. (2008). Weighted sampling for large-scale boosting. In Everingham, M., Needham, C. J., and Fraile, R., editors, *Proc. of the British Machine Vision Conference*, pp. 42.1–42.10, Leeds, UK. British Machine Vision Association.

[Karush, 1939] Karush, W. (1939). Minima of Functions of Several Variables with Inequalities as Side Constraints. Master's thesis, Dept. of Mathematics, Univ. of Chicago.

[Kearns and Valiant, 1994] Kearns, M. and Valiant, L. (1994). Cryptographic limitations on learning boolean formulae and finite automata. *Journal ACM*, 41(1):67–95.

[Kecman and Hadzic, 2000] Kecman, V. and Hadzic, I. (2000). Support vectors selection by linear programming. In *Proc. Intl. Joint Conf. on Neural Networks*, volume 5, pp. 193–198, Washington, DC.

[Kim et al., 2003] Kim, H.-C., Pang, S., Je, H.-M., Kim, D., and Bang, S. Y. (2003). Constructing support vector machine ensemble. *Pattern Recognition*, 36(12):2757 – 2767.

[Kimeldorf and Wahba, 1971] Kimeldorf, G. and Wahba, G. (1971). Some results on Tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications*, 33(1):82–95.

[Kuhn and Tucker, 1951] Kuhn, H. and Tucker, A. (1951). Nonlinear programming. In Neyman, J., editor, *Proc. of the Second Berkeley Symposium on Mathematical*

Statistics and Probability, pp. 481–492. University of California Press, Berkeley, CA.

[Kuncheva, 2004] Kuncheva, L. I. (2004). *Combining Pattern Classifiers: Methods and Algorithms*. Wiley, Hoboken, NJ.

[Kwok, 1999] Kwok, J. T. Y. (1999). Moderating the outputs of support vector machine classifiers. *IEEE Trans. Neural Networks*, 10(5):1018–1031.

[LeCun et al., 1995] LeCun, Y., Jackel, L. D., Eduard, H. A., Bottou, N., Cortes, C., Denker, J. S., Drucker, H., Sackinger, E., Simard, P., and Vapnik, V. (1995). Learning algorithms for classification: A comparison on handwritten digit recognition. In Oh, J. H., Kwon, C., and Cho, S., editors, *Neural Networks: The Statistical Mechanics Perspective*, pp. 261–276, Singapore. World Scientific.

[Li et al., 2008] Li, X., Wang, L., and Sung, E. (2008). AdaBoost with SVM-based component classifiers. *Engineering Applications of Artificial Intelligence*, 21:785–795.

[Liao et al., 2007] Liao, X., Li, H., and Carin, L. (2007). Quadratically gated mixture of experts for incomplete data classification. In *Proc. of the 24th International Conference on Machine Learning*, ICML '07, pp. 553–560, New York, NY. ACM Pres.

[Lima et al., 2009] Lima, N., Neto, A., and de Melo, J. (2009). Creating an ensemble of diverse support vector machines using adaboost. In *Proc. Intl. Joint Conf. on Neural Networks*, pp. 1802 –1806.

[Lu et al., 2006] Lu, J., Plataniotis, K., Venetsanopoulos, A., and Li, S. (2006). Ensemble-based discriminant learning with boosting for face recognition. *IEEE Trans. Neural Networks*, 17(1):166 –178.

[Lugosi and Vayatis, 2004] Lugosi, G. and Vayatis, N. (2004). On the Bayes-risk consistency of regularized boosting methods. *The Annals of Statistics*, 32(1):30–35.

[Lyhyaoui et al., 1999] Lyhyaoui, A., Martínez-Ramón, M., Mora, I., Váquez, M., Sancho, J.-L., and Figueiras-Vidal, A. R. (1999). Sample selection via clustering to construct support vector-like classifiers. *IEEE Trans. Neural Networks*, 10(6):1474–1481.

[MacKay, 2002] MacKay, D. J. C. (2002). *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY.

[Mallapragada et al., 2009] Mallapragada, P. K., Jin, R., Jain, A. K., and Liu, Y. (2009). Semiboost: Boosting for semi-supervised learning. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 31(11):2000–2014.

[Mannor and Meir, 2002] Mannor, S. and Meir, R. (2002). On the existence of linear weak learners and applications to boosting. *Machine Learning*, 48:219–251.

[Masnadi-Shirazi and Vasconcelos, 2011] Masnadi-Shirazi, H. and Vasconcelos, N. (2011). Cost-sensitive boosting. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 33(2):294 –309.

[Masnadi-Shirazi and Vasconcelos, 2007] Masnadi-Shirazi, H. and Vasconcelos, N. (2007). Asymmetric boosting. In *Proc. of the 24th International Conference on Machine Learning*, pp. 609–619, New York, NY. ACM Pres.

[Mason et al., 2000a] Mason, L., Bartlett, P. L., and Baxter, J. (2000a). Improved generalization through explicit optimization of margins. *Machine Learning*, 38:243–255.

[Mason et al., 2000b] Mason, L., Baxter, J., Bartlett, P., and Frean, M. (2000b). Functional gradient techniques for combining hypotheses. In Smola, A., Bartlett,

P., Schölkopf, B., and Schuurmans, D., editors, *Advances in Large Margin Classifiers*, pp. 221–246, Cambridge, MA. MIT Press.

[Mayhua-López et al., 2010] Mayhua-López, E., Gómez-Verdejo, V., and Figueiras-Vidal, A. R. (2010). Improving boosting performance with a local combination of learners. In *Proc. Intl. Joint Conf. on Neural Networks*, pp. 1983–1990, Barcelona.

[Mayhua-López et al., 2012] Mayhua-López, E., Gómez-Verdejo, V., and Figueiras-Vidal, A. R. (2012). Real adaboost with gate controlled fusion. *IEEE Trans. Neural Networks & Learning Systems*, 23(12):2003–2009.

[Mayhua-López et al., 2013] Mayhua-López, E., Gómez-Verdejo, V., and Figueiras-Vidal, A. R. (2013). Boosting ensembles with subsampled lpsvm learners. *Remitido a IEEE Trans. Neural Networks & Learning Systems*.

[Mease and Wyner, 2008] Mease, D. and Wyner, A. (2008). Evidence contrary to the statistical view of boosting. *Journal of Machine Learning Research*, 9:131–156.

[Meir et al., 2000] Meir, R., El-Yaniv, R., and Ben-David, S. (2000). Localized boosting. In *Proc. 13th Annual Conf. on Computational Learning Theory*, pp. 190–199, San Francisco, CA. Morgan Kaufmann.

[Meir and Rätsch, 2003] Meir, R. and Rätsch, G. (2003). An introduction to boosting and leveraging. In Mendelson, S. and Smola, A., editors, *Advanced Lectures on Machine Learning* (LNCS 2600), pp. 118–184, New York, NY. Springer.

[Michalski, 1980] Michalski, R. S. (1980). Pattern recognition as rule-guided inductive inference. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2(4):349–361.

[Moody and Darken, 1989] Moody, J. and Darken, C. J. (1989). Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1:281–294.

[Mosek, 2010] Mosek (2010). *The MOSEK Optimization Tools Manual Version 6.0 (Revision 66)*. MOSEK ApS, Copenhagen, Denmark.

[Omari and Figueiras-Vidal, 2013] Omari, A. and Figueiras-Vidal, A. R. (2013). Feature combiners with gate-generated weights for classification. *IEEE Trans. Neural Networks and Learning Systems*, 24(1):158 –163.

[Opelt et al., 2006] Opelt, A., Pinz, A., and Zisserman, A. (2006). A boundary-fragment-model for object detection. In *Proc. of the 9th European conf. on Computer Vision - Vol. Part II*, pp. 575–588, Berlin, Heidelberg. Springer-Verlag.

[Peng et al., 1996] Peng, F., Jacobs, R. A., and Tanner, M. A. (1996). Bayesian inference in mixtures-of-experts and hierarchical mixtures-of-experts models with an application to speech recognition. *J. of the American Statistical Association*, 91(435):953–960.

[Platt, 1991] Platt, J. C. (1991). Leaning by combining memorization and gradient descent. In Lippmann, R. P., Moody, J. E., and Touretzky, D. S., editors, *Advances in Neural Information Processing Systems 3*, pp. 714–720, San Mateo, CA. Morgan Kaufmann.

[Poor, 2010] Poor, H. (2010). *An Introduction to Signal Detection and Estimation*. Springer, Secaucus, NJ.

[Rangel et al., 2005] Rangel, P., Lozano, F., and García, E. (2005). Boosting of support vector machines with application to editing. In *Proc. 4th Intl. Conf. on Machine Learning and Applications*, p. 6.

[Rasmussen and Williams, 2005] Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, MA.

[Rätsch et al., 2002a] Rätsch, G., Mika, S., Scholkopf, B., and Muller, K.-R. (2002a). Constructing boosting algorithms from svms: an application to one-class classification. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(9):1184 – 1199.

[Rätsch et al., 2002b] Rätsch, G., Mika, S., and Warmuth, M. K. (2002b). On the convergence of leveraging. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Proc. Sys. 14*, pp. 487–494, Cambridge, MA. MIT Press.

[Rätsch et al., 1999] Rätsch, G., Onoda, T., and Müller, K. R. (1999). Regularizing adaboost. In Kearns, M., Solla, S., and Cohn, D., editors, *Proc. Advances in Neural Information Proc. Sys. 11*, pp. 564–570, Cambridge, MA. MIT Press.

[Rätsch et al., 2001] Rätsch, G., Onoda, T., and Müller, K. R. (2001). Soft margins for adaboost. *Machine Learning*, 42:287–320.

[Rätsch and Warmuth, 2005] Rätsch, G. and Warmuth, M. K. (2005). Efficient margin maximizing with boosting. *Journal of Machine Learning Research*, 6:2131–2152.

[Ripley, 1994] Ripley, B. D. (1994). Neural networks and related methods for classification. *J. Royal Statistical Society*, 56(3):409–456.

[Ripley, 1996] Ripley, B. D. (1996). *Pattern Recognition and Neural Networks*. Cambridge Univ. Press, Cambridge, UK.

[Rokach, 2010] Rokach, L. (2010). *Pattern Classification Using Ensemble Methods*. World Scientific, Singapore.

[Rudin et al., 2004] Rudin, C., Schapire, R. E., and Daubechies, I. (2004). Boosting based on a smooth margin. In *Proc. of the 17th Annual Conf. on Computational Learning Theory*, pp. 502–517.

[Rudin et al., 2007] Rudin, C., Schapire, R. E., and Daubechies, I. (2007). Analysis of boosting algorithms using the smooth margin function. *The Annals of Statistics*, 35(6):2723–2768.

[Ruiz and de Teruel, 2001] Ruiz, A. and de Teruel, P. E. L. (2001). Nonlinear kernel-based statistical pattern analysis. *IEEE Trans. Neural Networks*, 12(1):16–32.

[Schapire, 1990] Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 5:197–227.

[Schapire et al., 1998] Schapire, R. E., Bartlett, P., Freund, Y., and Lee, W. S. (1998). Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686.

[Schapire and Freund, 2012] Schapire, R. E. and Freund, Y. (2012). *Boosting: Foundations and Algorithms*. MIT Press, Cambridge, MA.

[Schapire and Singer, 1999] Schapire, R. E. and Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37:297–336.

[Schapire and Singer, 2000] Schapire, R. E. and Singer, Y. (2000). Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2-3):135–168.

[Schölkopf et al., 1997] Schölkopf, B., Kah-Kay, S., Burges, C. J. C., Girosi, F., Niyogi, P., Poggio, T., and Vapnik, V. (1997). Comparing support vector machines with Gaussian kernels to radial basis function classifiers. *IEEE Trans. Signal Processing*, 45(11):2758–2765.

[Schölkopf and Smola, 2002] Schölkopf, B. and Smola, A. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA.

[Schwenk and Bengio, 1997] Schwenk, H. and Bengio, Y. (1997). Adaboosting neural networks. In Gerstner, W., Germond, A., Hasler, M., and Nicoud, J. D., editors, *Proc. 7th International Conf. on Artificial Neural Networks* (LNCS 1327), pp. 967–972, Berlin. Springer.

[Schwenk and Bengio, 2000] Schwenk, H. and Bengio, Y. (2000). Boosting neural networks. *Neural Computation*, 12:1869–1887.

[Sharkey, 1999] Sharkey, A. J. (1999). (ed.), *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*. Springer-Verlag, London, UK.

[Shawe-Taylor and Cristianini, 2004] Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY.

[Shen and Li, 2010] Shen, C. and Li, H. (2010). On the dual formulation of boosting algorithms. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32(12):2216 –2231.

[Shin and Cho, 2002] Shin, H. J. and Cho, S. (2002). Pattern selection for support vector classifiers. In *Proc. 3rd Intl. Conf. on Intelligent Data Engineering and Automated Learning*, pp. 469–474, Manchester, UK.

[Shin and Cho, 2003a] Shin, H. J. and Cho, S. (2003a). Fast pattern selection for support vector classifiers. In *Proc. 7th Pacific-Asia conf. Advances in Knowledge Discovery and Data Mining* (LNAI 2637), pp. 376–387, Seoul, Korea.

[Shin and Cho, 2003b] Shin, H. J. and Cho, S. (2003b). How many neighbors to consider in pattern pre-selection for support vector classifiers? In *Proc. Intl. Joint Conf. on Neural Networks*, pp. 565–570, Portland, OR.

[Shin and Cho, 2007] Shin, H. J. and Cho, S. (2007). Neighborhood property-based pattern selection for support vector machines. *Neural Computation*, 19:816–855.

[Sun and Yao, 2010] Sun, P. and Yao, X. (2010). Sparse approximation through boosting for learning large scale kernel machines. *IEEE Trans. Neural Networks*, 21(6):883–894.

[Tieu and Viola, 2004] Tieu, K. and Viola, P. (2004). Boosting image retrieval. *Int. Journal of Computer Vision*, 56(1-2):17–36.

[Torralba et al., 2004] Torralba, A., Murphy, K. P., and Freeman, W. T. (2004). Sharing features: efficient boosting procedures for multiclass object detection. In *Proc. of the 2004 IEEE Comput. Society Conf. on Comput. Vision and Pattern Recognition*, pp. 762–769, Washington, DC. IEEE Computer Society.

[Valiant, 1984] Valiant, L. G. (1984). A theory of the learnable. *Commun. ACM*, 27:1134–1142.

[Van Trees, 2004] Van Trees, H. (2004). *Detection, Estimation, and Modulation Theory*. Number Vol. I. Wiley, New York, NY.

[Vapnik, 1995] Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer, New York, NY.

[Vapnik, 1998] Vapnik, V. (1998). *Statistical Learning Theory*. Wiley, New York, NY.

[Vapnik and Chervonenkis, 1971] Vapnik, V. N. and Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280.

[Viola and Jones, 2001] Viola, P. and Jones, M. (2001). Robust real-time object detection. In *Intl. Journal of Computer Vision*.

[Viola and Jones, 2004] Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *Int. Journal of Computer Vision*, 57(2):137–154.

[Warmuth et al., 2008a] Warmuth, M. K., Glocer, K. A., and Rätsch, G. (2008a). Boosting algorithms for maximizing the soft margin. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T., editors, *Advances in Neural Information Processing Systems 20*, Vancouver, Canada. Curran.

[Warmuth et al., 2008b] Warmuth, M. K., Glocer, K. A., and Vishwanathan, S. V. (2008b). Entropy regularized lpboost. In *Proc. of the 19th intl. conference on Algorithmic Learning Theory*, pp. 256–271, Berlin, Heidelberg. Springer-Verlag.

[Warmuth et al., 2006] Warmuth, M. K., Liao, J., and Rätsch, G. (2006). Totally corrective boosting algorithms that maximize the margin. In Cohen, W. and Moore, A., editors, *Proc. Intl. Conf. on Machine Learning*, pp. 1001–1008.

[Wei et al., 2010] Wei, T., Qin, Z., Cao, X., and Leng, B. (2010). A boosting method based on SVM for relevance feedback in content-based 3D model retrieval. In *Proc. 2nd Intl. Conf. on Software Engineering and Data Mining*, pp. 517–522.

[Wickramaratna et al., 2001] Wickramaratna, J., Holden, S., and Buxton, B. (2001). Performance degradation in boosting. In Kittler, J. and Roli, F., editors, *Multiple Classifier Systems* (LNCS 2096), pp. 11–21. Springer, Berlin.

[Wu et al., 2009] Wu, W., Yanan, Z., and Linlin, W. (2009). An adaboost algorithm with svm based on nonlinear decision function. In *Proc. Intl. Conf. on Computational Intell. and Natural Computing*, pp. 22–25.

[Yuksel et al., 2012] Yuksel, S., Wilson, J., and Gader, P. (2012). Twenty years of mixture of experts. *IEEE Trans. Neural Networks Learning Sys.*, 23(8):1177 –1193.

[Zhang and Zhang, 2008] Zhang, C.-X. and Zhang, J.-S. (2008). Rotboost: A technique for combining rotation forest and adaboost. *Pattern Recognition Letters*, 29(10):1524–1536.

[Zhou et al., 2002] Zhou, W., Zhang, L., and Jiao, L. (2002). Linear programming support vector machines. *Pattern Recognition*, 35(12):2927 – 2936.

[Zhu et al., 2004] Zhu, J., Rosset, S., Hastie, T., and Tibshirani, R. (2004). 1-norm support vector machines. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*, pp. 49–56, Cambridge, MA. MIT Press.